DEFENCE R&D DÉFENSE

A Simulation of P_MUL (An Application for Multicast Messaging under EMCON Restriction)

Claude Bilodeau and Sarah Dumoulin
Communications Research Centre Canada

CRC Communications Research Centre Centre de recherches sur les communications

The work described in this document was sponsored by the Department of National Defence under Work Unit 2KN14

Defence R&D Canada
DEFENCE RESEARCH ESTABLISHMENT OTTAWA

TECHNICAL REPORT
DREO TR 2000-048
CRC Report No. 2000-005
June 2000

DISTRIBUTION STATEMENT A
Approved for Public Release
Distribution Unlimited

National Defence Défense nationale

Canada

DEFENCE R&D DÉFENSE

A Simulation of P_MUL (An Application for Multicast Messaging under EMCON Restriction)

Claude Bilodeau and Sarah Dumoulin
Broadband Network Technologies Branch
Communications Research Centre Canada

CRC Communications Research Centre / Centre de recherches sur les communications

The work described in this document was sponsored by the Department of National Defence under Work Unit 2KN14

DEFENCE RESEARCH ESTABLISHMENT OTTAWA

TECHNICAL REPORT
DREO TR 2000-048
CRC Report No. 2000-005
June 2000

A Simulation of P_MUL
(An Application for Multicast Messaging under EMCON Restriction)

Claude Bilodeau and Sarah Dumoulin
Communications Research Centre Canada

The work described in this document was sponsored by the Department of National Defence under Work Unit 2KN14

Defence Research Establishment Ottawa

Technical Report

DREO TR 2000-048
CRC Report No. 2000-005
June 2000

Abstract

P_MUL is an application layer reliable multicast protocol specifically designed for use over narrowband networks where nodes may be operating in EMCON (Emission Control) or *Radio silence* mode. The protocol is specified in the Allied Communication Publication (ACP) 142. The Communications Research Centre (CRC) has created a discrete-event simulation model of P_MUL in order to characterise its performance under a variety of operating conditions. Simulation scenarios were created to examine the effectiveness of P_MUL in both noise-free and noisy channel environments. The scenarios focus on the impact of protocol parameters on the results obtainable by P_MUL.

This report describes the CRC model of P_MUL and outlines the simulation scenarios studied. Analysis of simulation results is provided, weaknesses of the protocol are identified, and optimal values for certain protocol parameters are suggested.

Résumé

P_MUL est une application de messagerie multipoint et fiable spécifiquement conçue pour être utilisée sur les réseaux à bande passante étroite dont les nœuds fonctionnent parfois en mode EMCON (contrôle d'émission ou mode *silence radio*). Les spécifications du protocole apparaissent dans la Publication interalliées sur les communications (ACP) 142. Le Centre de recherches sur les communications (CRC) a mis au point un modèle de simulation à événements discrets en vue de préciser la performance de P_MUL dans diverses conditions. Plusieurs scénarios de simulation ont été élaborés afin d'établir l'efficacité du protocole dans les transmissions avec ou sans bruit sur réseau. Ces scénarios mettent en relief l'incidence des paramètres du protocole sur les résultats obtenus avec P_MUL.

Le présent rapport décrit le modèle du CRC et les simulations effectuées. On y trouvera aussi l'analyse des résultats, les points faibles du protocole et les valeurs optimales de certains paramètres.

Executive Summary

Background

Reliable multicast protocols enable a sender to reliably transfer messages simultaneously to a group of one or more receivers. Multicast routers are used to ensure that packets are not sent more than once over links that may be shared by one or more destinations. The resulting reduction in the bandwidth required for message transfer can be significant for low-bandwidth networks.

P_MUL is a reliable multicast protocol originating from the Communications Systems Network Interoperability (CSNI) Project. It was designed to support the X.400 Message Handling Systems, including support for operation under EMCON conditions. P_MUL has been specified in the draft Allied Communication Publication (ACP) 142. The first release of the standard is expected before the end of 2000. An Internet-Draft of the protocol is also under development.

The Study

CRC developed a discrete-event simulation model of the P_MUL protocol in order to investigate its performance under a variety of network conditions.The model is based on the Optimized Network Engineering Tools (OPNET®) software. This report presents an overview of the CRC model, including a description of the model architecture, the simulation parameters, and the available output metrics.

The report also provides a study of P_MUL through an analysis of the simulation results. The simulation scenarios presented investigate the impact of various protocol parameters on P_MUL's performance. The first set of scenarios are concerned with the operation of P_MUL under ideal (error-free) network channel conditions. The simulation results demonstrate that when operating in a noiseless environment, P_MUL is a quiet and efficient protocol. The results also indicate that under certain conditions P_MUL may pollute the network with unnecessary retransmissions. The use of different protocol parameters to effect some level of congestion avoidance and congestion control (CA/CC) is examined for static system configurations. If *a priori* knowledge of the network traffic conditions is available, the parameters are found to be effective in reducing the number of unneeded packets transmitted by P_MUL. In practice, a more dynamic form of CA/CC may be necessary but the CRC model would need to be modified to further study the issue.

The second set of simulation scenarios examine the usability of P_MUL over noisy network channels. P_MUL's performance under noisy conditions is very different from its performance under noiseless conditions. Once errors are introduced, the study shows that the protocol looses some of its robustness and becomes vulnerable to the loss of control packets. The network can at times become polluted with a large number of unneeded packets and often fail to properly terminate message transfers. The study also indicates

that packet size is the most important parameter effecting P_MUL's message tranfer reliability. Even when the packet size is optimized, P_MUL is rather limited in its capability to deliver messages in a reasonable length of time at high bit error rates (e.g. above 10^{-3} errors per bit). The use of erasure coding to improve P_MUL's reliability was investigated, but although it was able to significantly improve the performance at low bit error rates the results were unconvincing at high bit error rates. It would be necessary to compare P_MUL to other reliable transport protocols to better assess the performance obtained in similar conditions.

Military Significance

P_MUL is specifically designed for military networks. In particular, it is envisioned that P_MUL will be deployed in narrowband network environments. As a result, emphasis is placed on keeping the overhead required by the protocol to a minimum in order to conserve bandwidth. When operating over a noiseless network P_MUL is a highly efficient protocol, requiring only one acknowledgment from every receiver.

Furthermore, nodes operating under EMCON (*Radio silence*) conditions are natively supported by the protocol. A separate mode of operation is available to reliably transfer messages to nodes operating under EMCON conditions, and the protocol is able to wait for long periods for acknowledgements from EMCON nodes. Because of its traffic saving capacity and half-duplex operating capability, P_MUL can play a strategic role within military networks.

Suggestions for Future Research Work

Some of the open issues that require further study include:

i. improvement of the protocol's ability to recover from the loss of control packets;

ii. implementation of a dynamic CA/CC scheme to improve performance in heterogeneous network environments;

iii. the usefulness of increasing the amount of feedback from the receivers;

iv. the ability of the protocol to avoid acknowledgement (ACK) implosion at the sender.

Sommaire

Introduction

Les systèmes de messagerie multipoint fiable permettent à l'expéditeur d'envoyer des messages simultanément à un ou plusieurs destinataires en toute sécurité. On se sert de routeurs multipoint pour s'assurer que les paquets de données évitent de traverser plus d'une fois un parcours susceptible d'aboutir à d'autres destinations. La réduction de la largeur de bande nécessaire à la transmission du message qui en résulte peut s'avérer fort intéressante pour les réseaux à bande passante étroite.

P_MUL est un protocole multipoint fiable mis au point dans le cadre du Projet d'interopérabilité des réseaux de systèmes de télécommunications (CSNI). Il a été conçu à l'ori-gine pour les systèmes de messagerie X.400 et en permettait l'exploitation en mode EMCON. Les spécifications de P_MUL apparaissent dans la Publication interalliées sur les communications (ACP) 142. La première version de la norme devrait être disponible avant la fin de l'an 2000. Il existe également un projet de protocole Internet.

L'étude

Le CRC a créé un modèle de simulation à événements discrets en vue d'étudier l'efficacité de P_MUL en réseau dans diverses situations. Le modèle fait appel au logiciel OPNET© (Optimized Network Engineering Tools). Le rapport qui suit donne une vue d'ensemble du modèle, notamment son architecture, les paramètres de la simulation et les données de sortie exploitables.

On y trouvera aussi une étude de P_MUL articulée sur l'analyse des résultats de simulation. Les scénarios présentés apportent des précisions sur l'incidence de différents paramètres du protocole sur son efficacité. Dans la première série de simulations, on suppose que P_MUL fonctionne dans des conditions de transmission idéales en réseau (aucune erreur). Les résultats indiquent que P_MUL est un protocole aussi silencieux qu'efficace dans un environnement sans bruit. Ils démontrent également que dans certaines conditions, P_MUL peut polluer le réseau avec des retransmissions inutiles. Les auteurs examinent comment certains paramètres du protocole permettent, pour des configurations statiques du système, un certain degré d'évitement et de réduction de la congestion. Si les débits d'utilisation du réseau sont connus *a priori*, les paramètres permettent de réduire le nombre de paquets inutiles envoyés par P_MUL. Dans la pratique, il vaudrait la peine de recourir à une méthode plus dynamique d'évitement et de réduction de la congestion, mais il sera nécessaire de modifier le modèle du CRC pour approfondir la question.

La deuxième série de simulations porte sur le rendement de P_MUL lorsque les voies de communication sont bruyantes. La performance de P_MUL dans de telles conditions diffère considérablement de celle observée en l'absence de bruit. Lorsqu'on introduit des erreurs, l'étude révèle que le protocole n'est plus aussi résistant et devient vulnérable à la

perte des paquets de données de contrôle. Il arrive que le réseau soit pollué par un grand nombre de paquets inutiles et les connexions des messages sont souvent mal terminées. L'étude indique également que la taille du paquet constitue le paramètre le plus important dont dépend la fiabilité des transmissions. L'aptitude de P_MUL à relayer les messages dans un délai raisonnable à un taux d'erreur binaire élevé (à savoir, supérieur à 10^{-3} erreurs par bit) est assez restreinte, même quand les paquets ont la taille optimale. On a essayé de déterminer si l'usage d'un code correcteur d'erreurs à oblitération (erasure coding) accroîtrait la fiabilité du protocole, mais en dépit d'une amélioration sensible à faible taux d'erreur binaire, les résultats restent décevants aux taux d'erreur élevés. Il faudrait comparer P_MUL à d'autres protocoles de transmission pour mieux évaluer sa performance dans des conditions similaires.

Signification pour les communications tactiques

P_MUL est spécifiquement conçu pour les réseaux tactiques. On envisage notamment de s'en servir avec les réseaux à bande passante étroite. Pour préserver la plus grande largeur de bande possible, on tente donc de maintenir les bits supplémentaires qu'exige le protocole au strict minimum. Avec un réseau sans bruit, P_MUL s'avère d'une grande efficacité et ne demande qu'un accusé de réception par destinataire.

Par ailleurs, le protocole est naturellement fait pour être utilisé avec les nœuds qui fonctionnent en mode EMCON (*silence radio*). Un mode distinct et fiable permet l'envoi de messages aux nœuds qui opèrent en EMCON et le protocole peut attendre longtemps, si nécessaire, l'accusé de réception venant de ces nœuds. À cause des économies de bande passante qu'il permet de réaliser et de son exploitation en mode demi-duplex, P_MUL peut jouer un rôle important dans les communications tactiques.

Suggestions de travaux de recherche

Quelques-unes des défaillances qui demandent une attention particulière, incluent:

1. amélioration de la résistance du protocole afin de poursuivre l'acheminement des messages malgré la perte des paquets de données de contrôle;
ii. introduction d'une méthode dynamique d'évitement et de réduction de la congestion en vue d'améliorer la performance du protocole dans les réseaux hétérogènes;
iii. utilité d'intensifier le degré de rétroaction des destinataires;
iv. aptitude du protocole à éviter l'implosion des accusés de réception (ACK) du côté de l'expéditeur.

DREO TR 2000-048, CRC-RP-2000-005, *Une simulation sur P_MUL (Une application de messagerie multipoint pour les communications tactiques en mode EMCON) (en anglais),* Claude Bilodeau et Sarah Dumoulin, CRC, VPNT-RNS, Ottawa

Table of Contents

List of Figures

List of Tables

1.0 Introduction

P_MUL is an application layer protocol, in some way similar to the P1 protocol, defined in the X.400 Recommendations [1], or the Simple Mail Transfer Protocol (SMTP) protocol, defined in the RFC 821 [2]. P_MUL supports very efficient one-to-n communication services over heterogeneous networks. It takes advantage of a multicast communication service to transfer messages between different nodes on a single multicast network under both normal, meaning full duplex oriented communication conditions, and under Emission control (EMCON) conditions. EMCON or *Radio Silence* condition means that a receiving node is able to receive messages, but cannot acknowledge received messages for a relatively long time (e.g. hours or days). Because of its traffic saving capacity and half-duplex operating capability, P_MUL can play a strategic role within military networks.

P_MUL is one of the results of the Communications Systems Network Interoperability (CSNI) Project [3]. During Phase 2 of the project, a specification was developed to support the X.400 Message Handling Systems (MHS) with multicast message distribution and message transfer under EMCON condition. The specification was validated by an implementation which was tested between laboratories within the cooperating nations. It was later demonstrated in different international and national trials. Based on this work, a working group of the Combined Communications Electronics Board (CCEB[1]) started to draft the Allied Communication Publication (ACP) 142 [4] in 1998. The first release of the standard is expected before the end of 2000. The ACP draft itself is largely based on an Internet-Draft [5] produced by Riechmann, a researcher at FGAN (Germany) who took part in the CSNI project.

The Communications Research Centre (CRC) was one of the organisations participating in the CSNI project and is a member of the ACP drafting team. Even though the demonstration testbeds led to some interesting results with regard to the performance of P_MUL in heterogeneous networks, CRC felt that the capabilities of P_MUL needed to be further investigated by other means. The approach taken is to develop a discrete-event simulation model for conducting non-real-time analysis of P_MUL. This report presents the model and its capabilities as well as some preliminary performance data of P_MUL — such as messaging throughput, delivery time, overhead and bit-error tolerance — for various network topologies and simulated conditions. This is believed to be the first study on P_MUL reported in the open literature.

1. The CCEB is a mutual cooperation organisation with membership from Australia, Canada, New Zealand, the United Kingdom, and the United States, who's purpose is to develop information exchange standards and protocols to support interoperability between the member nations. The standards are also widely used by other nations and organisations such as the North Atlantic Treaty Organization (NATO).

2.0 Overview of the P_MUL Protocol

P_MUL allows an application to transfer messages between a sender and one or more receivers through the use of a multicast routing protocol such as the Multicast Open Shortest Path First (MOSPF) protocol or the Protocol Independent Multicast (PIM) protocol. A multicast router will ensure that the message is not duplicated or uselessly retransmitted over links that may be shared by two or more destinations. The gain achieved by reducing the number of packets that must be sent to reach a group of destinations can be significant for low-bandwidth networks.

To make use of a multicast routing protocol, multicast groups must first be created. P_MUL groups may either be set up manually, if there are few well-established multicast groups, or dynamically, when the management of numerous groups starts to be a problem. In both cases, the set up process begins by selecting a group address, albeit the ACP 142 has yet to specify the exact procedure. This could, for instance, be done manually by supplying P_MUL with a predefined group address, or automatically by letting P_MUL choose randomly an address from an available group address space. In any case, once an address has been selected, the address needs to be made globally known by sending a REQUEST_PDU (Request Protocol Data Unit) to each possible transmitter on the network. After waiting for a predefined time interval, the node requesting ownership of an address will assume that the address is legitimate if no transmitter in the network has rejected its request by returning a REJECT_PDU. This could happen if an address is already in use somewhere in the network and the sender did not know about it. In such a case, a well-behaved sender will renege its request by sending a RELEASE_PDU and repeat the set up process by selecting a new group address. If the waiting period happened to be silent (without any response from the network) the sender can de facto use the address and inform the receiving nodes of the existence of the new multicast group. This is done by sending a series of ANNOUNCE_PDUs listing the members who should join the multicast group at the newly assigned address.

Once the multicast group has been set up, the sender starts the message transfer phase by sending a series of ADDRESS_PDUs containing the addresses of all of the intended recipients. Every recipient will create a message entry for the new message once it receives the ADDRESS_PDU containing its address.

The message transfer phase continues with the sender transmitting the message itself in the form of DATA_PDUs. One or more retransmissions will take place if any one of the recipients fails to acknowledge correct receipt of the entire message. Furthermore, the sender will constantly adapt its retransmission mode over the course of a message transfer according to the operating condition of the pending recipients. There are two possible retransmission modes:

1. The sender only enters the *EMCON retransmission mode* if all the pending recipients are operating in EMCON conditions. In this mode, the sender periodically transmits all ADDRESS_PDUs and DATA_PDUs up to a predefined maximum number of retransmissions.

2. In the *non-EMCON retransmission mode*, the sender periodically transmits the ADDRESS_PDUs and DATA_PDUs for which no acknowledgement has been received.

Conversely, receivers are subjected to two possible operating conditions:

1. Receivers operating in *EMCON conditions* cannot transmit any acknowledgement PDUs (ACK_PDUs) to the sender. This forces the receivers to delay acknowledgement of the DATA_PDUs they receive until they leave the EMCON conditions.
2. A receiver operating in *non-EMCON conditions* unicasts an ACK_PDU towards the sender when the current transmission of the message has ended. If, over the course of a message transfer, the receiver detects that it has missed more than some predefined number of DATA_PDUs it must unicast to the sender an ACK_PDU listing all PDUs that it has missed.

When the sender receives an ACK_PDU from a receiver listing no missing DATA_PDUs it removes the recipient's address from the ADDRESS_PDU set and informs the application that the recipient has successfully acknowledged the message. Similarly, when a receiver has received the entire message it passes the message to the application. However, a receiver will not destroy the message entry until it receives an acknowledgement to its final ACK_PDU from the sender. This should come in the form of an ADDRESS_PDU set in which its address does not appear.

The receiver could be operating in EMCON conditions when the message reception is completed. In such a case, the final ACK_PDU is held until the receiver exits EMCON conditions.

The message transfer will be aborted if the message expires before the sender receives the final ACK_PDUs from all receivers. If this occurs, the sender will consider that all pending nodes in non-EMCON conditions have failed to receive the message and will inform the application accordingly. The sender then multicasts a DISCARD_PDU, which causes the following to happen:

1. Receivers in non-EMCON conditions will discard all DATA_PDUs received and destroy the message entry upon receiving the DISCARD_PDU.
2. Receivers in EMCON conditions who have not yet received the entire message will form an ACK_PDU listing missing PDUs to be sent when they exit EMCON and discard all DATA_PDUs received so far. Receivers in EMCON conditions who have already received the entire message will disregard the DISCARD_PDU.

The sender always waits for a response from nodes in EMCON conditions as they may have successfully received the message.

3.0 Modelling Information

In any modelling project, models are limited for practical reasons to representing only certain aspects of the system of interest. In the case of P_MUL, some kind of test environment is needed to conduct performance characterisation of the protocol. The P_MUL model has therefore been wrapped in a modelling environment that allows a wide range of performance tests, some approximating a real environment, others being rather synthetic or not resembling any existing real world condition. Most importantly, the model is sufficiently detailed to conduct a thorough analysis and should provide interesting and useful benchmarks for countless operating conditions.

This section describes the simulation architecture, which includes the P_MUL model and its main characteristics.

3.1 Simulation Architecture

The discrete-event simulation architecture developed by the CRC is based on the Optimized Network Engineering Tools (OPNET®) software produced by OPNET Technologies Inc.. OPNET is a sophisticated workstation-based environment supporting the modelling and performance-evaluation of communication protocols and distributed network systems. It is designed around a dynamic, event-scheduled simulation kernel which is accessible through a large number (over 275) of user-callable procedures.

The simulation architecture is comprised of several components that behave in some sense like a real system: a Message driver, the P_MUL protocol model, a UDP/IP (User Datagram Protocol/Internet Protocol) protocol stack, a static router, a generic data link layer (DLL) and two types of communication links: point-to-point and broadcast. In addition, some other components were included to generate background traffic on the network links and for sending messages in unicast TCP/IP (Transmission Control Protocol/Internet Protocol) mode for performance comparison.

Except for the communication links, all these components originate from custom development of OPNET nodes. The architecture consists of four basic types of nodes:

1. *backbone*, to represent network routers in the backbone network;
2. *busnode*, to represent a host node (either a T-node or an R-node) on a broadcast network;
3. *host_node*, to represent a host node (either a T-node or an R-node) for point-to-point network; and,
4. *bus_Rx_only*, to model a node where a back-channel is used to acknowledge messages coming from a broadcast network.

These node models are architectural components much like building blocks that can be interconnected in may different ways to simulate and analyse various network topolo-

gies. Several simulation runs can be setup to investigate particular aspects of the P_MUL protocol without any software change.

3.1.1 Modules and their Functionality

The functionality of each node model varies somewhat but the busnode model is perhaps the one that best describes the most capabilities of all of them. For that reason, it will be used here as an example to describe the various modules and the structure pf most nodes. The busnode model is shown in Figure 1.

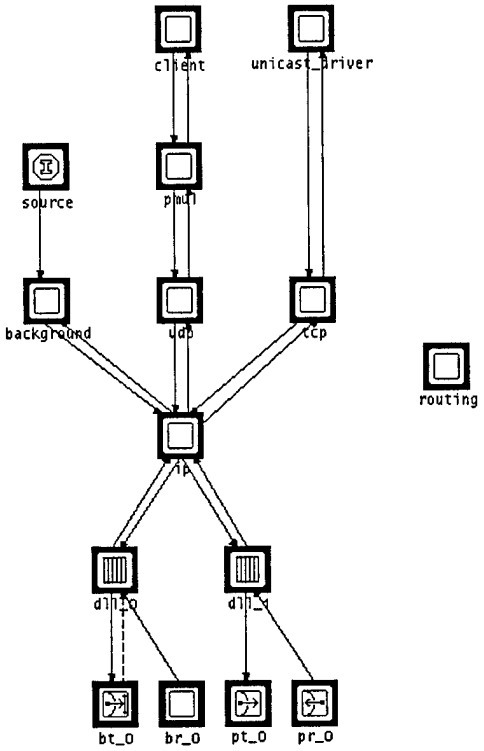


FIGURE 1. The P_MUL module in a busnode node

Essentially, the node implements the protocol stack shown in Figure 2. The modules in Figure 1 are described layer by layer in the next subsections.

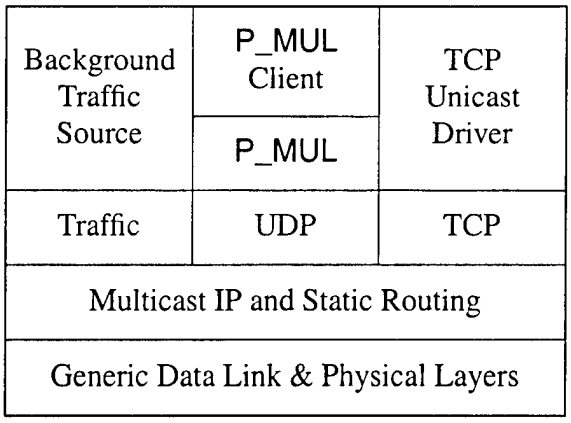


FIGURE 2. Protocol stack of the busnode model

3.1.1.1 Application Layer

`Client`

The `Client` module is the main source of messages for the simulation. Through model attributes of the `Client` module, the user can schedule the sending in either multicast or unicast mode of a message of any given size towards any destination. Optionally, the name of a script file can also be specified if more than one message must be sent.

`Unicast_driver`

As for the `Client` module, the `Unicast_driver` initiates the sending of messages throughout the network. However, the `Unicast_driver` module can only send unicast messages over TCP/IP. The `Client` and the `Unicast_driver` modules share the same script file for sending more than one message when necessary. Note that the `Unicast_driver` connects directly to the `tcp` module and does not allocate any of the overhead that would normally be present in applications such as those for the MHS, e.g., P1 over TP0 for X.400 based MHS or SMTP for RFC822 based MHS. A fair comparison of the P_MUL performance against TCP would require that a portion of the actual message size be considered as application protocol overhead.

`Source`

This module is a default OPNET packet generator. It can be used to generate packets of different sizes and at various production rates.

`pmul`

The `pmul` module is where the P_MUL model resides. The module performs most, if not all, the functions of the P_MUL protocol as defined in the ACP 142, including some features only mentioned in the Internet-Draft. The model is capable of managing several concurrent ingoing and outgoing messages. It supports the dynamic installation of multicast groups as well as static group configuration.

3.1.1.2 Transport Layer

`udp`

The `udp` module is the connectionless transport protocol used by P_MUL to transmit and receive messages. The UDP protocol model currently supports only one client, i.e. P_MUL.

`tcp`

The `tcp` module is the connection-oriented transport protocol used by the `Unicast_driver` to transmit and receive messages in non-P_MUL unicast mode. Only selected functions of the TCP protocol are implemented.

`background`

The `background` module provides a means of loading the links with traffic having a well defined probability distribution. The module encapsulates the packets generated by the `Source` module and sends them out via the `ip` module towards a specified (usually adjacent) node. A peer module at the destination node accepts the packets from the `ip` module and destroys them.

3.1.1.3 Network Layer

`ip`

The `ip` module implements selected functions of the IP protocol. These functions include encapsulation of the transport protocol data units and decapsulation of the IP datagrams, as well as unicast and multicast forwarding, including packet replication. Since P_MUL limits the size of the packets, no fragmentation and reassembly has been implemented in `ip`.

`routing`

This module performs "static" routing i.e. forwarding tables for all nodes in the network are created just once at initialisation, stored in global memory and updated as needed during a simulation run. There is no exchange of routing information over the network links. Unicast and multicast forwarding tables are computed using the shortest paths from all nodes to all other nodes. This includes processing any input file of predefined multicast group addresses and creating a global multicast address file.When a multicast group address is created or destroyed and tables need to be updated, the `routing` module in the node that is or was the owner of the address in question updates the forwarding tables for all other nodes.

3.1.1.4 Link Layer

`dll`

The `dll` module includes two generic data link models, one for point-to-point links and one for bus (broadcast) links. In both cases, packets are sent on the link only once, there is no retransmission. The model supports both bidirectional and unidirectional links.

The point-to-point data link layer protocol model is the simplest. Packets are queued in infinite numbers if necessary while waiting for the link to become free. The packet at the head of the queue is encapsulated then transmitted at the nominal rate set for the link. The encapsulation consists of adding overhead bits to the IP packet in the amount specified by two attributes to the `pmul_ptp` link model. One attribute specifies the data link protocol header size and is taken to be fixed for every transmitted packet. The second attribute specifies the average amount of aggregated overhead introduced by other characteristics of the link protocol, characteristics such as those associated with any Automatic Repeat reQuest (ARQ) scheme, link-level Forward Error Correction (FEC) scheme, etc. that may be present in the type of link intended for the simulation.

The broadcast data link layer model is somewhat more complex. The link overhead specified by the `pmul_bus` link model attributes is added in the same way as for the point-to-point link. The model also supports infinite queue length and a predefined transmission rate. However, being a shared medium, the broadcast model requires medium access control (MAC).

The MAC scheme that has been devised for this study is simple but cannot be physically implemented in practice. It consists of setting up a transmission priority list in global memory where every host sharing the medium can subscribe no more than one packet to the list at a time. Any subscription is entered at the tail of the list and if a host has more packets to send, it must enqueue them until the list no longer contains the previously subscribed packet. When the bus becomes free, the head of the list determines which host is granted access to the bus. This collision-free scheme maximises the channel utilization while providing some level of

fairness among the hosts wishing to access the channel. It has the advantage of providing an upper bound on the throughput of P_MUL when it is implemented in a multiple access channel environment. The scheme ensures that the link will not be idle if there are packets waiting to be transmitted.

Note that the `dll` module is designed so that it should be possible to include a more complex model of a specific data link layer protocol at a later date, should such a modification be desired.

3.1.1.5 Physical Layer

`pt, pr, bt, br`

The point-to-point transmitter `pt`, the point-to-point receiver `pr`, the bus transmitter `bt` and the bus receiver `br` are all standard OPNET channel modules. They are the entry points in a node for packets received from the links and the exit points for packets forwarded to the links.

3.1.2 Channel Models

`pmul_ptp, pmul_bus`

The point-to-point channel model `pmul_ptp` and the bus channel model `pmul_bus` are simple derivatives of the standard OPNET channel models consisting of multi-stage computational pipelines. These models are associated with every instance of link objects used to interconnect nodes in a simulation topology. Each model is characterised by a number of attributes, including:

- a *data rate* attribute, to set the nominal channel (link transmission) capacity. This attribute along with the length of the packet determines the transmission delay, i.e., the time that the transmitter requires to completely process and transmit the packet.

- a *propagation delay* attribute, to account for the delay incurred by packets while travelling through the transmission medium. The `pmul_ptp` model supports any amount of propagation delay whereas the `pmul_bus` model does not currently support any amount of propagation delay. This limitation is not imposed by the MAC design itself; this capability has simply not been implemented yet.

- a *ber* attribute, to establish the bit-error-rate observed over the transmission channel. Given this error rate and the number of bits in the packet, the model stochastically injects bit errors in the packet. White noise is assumed which means that there is an equal probability of an error in each bit position and an independence of errors in different positions (uniform error distribution).

3.2 Bit-Error Handling

There are two levels of error correction in the model. They are both independent so either one of them or the two together can be activated before running a simulation. The first error correction level is performed at the link layer whereas the second is performed by the P_MUL protocol.

3.2.1 Generic Link Error-Control Coding Model

The first error correction level is performed at the link layer. When a packet is received at a node the model will accept the packet if the number of bit errors is less than a specified link error threshold, otherwise the packet is discarded. The threshold, expressed in bits, is the sum of two components:

1. a fixed number of acceptable errors (`ecc_abs_link`); and,
2. a proportional number of acceptable errors (`ecc_rel_link`) derived from the length of the packet.

The reason for having two components is to distinguish the error correction capability as it applies to the frame bits and the data bits respectively. The former is usually more robust and is of a fixed size whereas the second varies with the size of the payload.

3.2.2 Erasure Coding Model

The second error correction level is performed by the P_MUL protocol. Both the ACP 142 and the Internet-Draft do not specify any error correction code for P_MUL. The coding approach developed for this study is one of many options to add some robustness to the delivery of messages by P_MUL. Being a research area that requires further investigation, the encoding scheme presented herein should not be taken as being the preferred or only approach to use for improving the reliability of P_MUL.

The forward error correction (FEC) technique used in the model is erasure correction. The technique is described in [6][7] and has been suggested to the authors by Riechmann [8].

Rizzo [6] describes the technique in the following terms.

> The key idea behind erasure codes is that k blocks of source data are encoded at the sender to produce n blocks of encoded data, in such a way that any subset of k encoded blocks suffices to reconstruct the source data. Such a code is called an (n,k) code and allows the receiver to recover from up to n-k losses in a group of n encoded blocks.

There are many ways of adapting an (n,k) code to a message of length L. In P_MUL, `MPDU_SIZE` is a predefined protocol parameter specifying the size of the largest protocol data unit (PDU) that can be sent to the transport layer for transmission over the multicast network.

If a source data blocks is limited in size to fit into a P_MUL DATA_PDU, the maximum block size for encoding the source data is equal to MPDU_SIZE less the space reserved for the DATA_PDU header bits, i.e.:

$$\text{Maximum Erasure Block Size} = \texttt{MPDU\_SIZE} - \texttt{DATA\_PDU\_HEADER\_SIZE}$$

Doing so guarantees that a source data block will fit in a single DATA_PDU. An encoded message will therefore consist of up to n DATA_PDUs. The receiver will need to correctly detect only k of those DATA_PDUs. Note that it is conceivable to spread a source block into several DATA_PDUs but this would increase the complexity of the scheme.

If the message is large, the message must either be broken into a series of short messages and transmitted in several successive pieces or else the coding parameters must be changed to suit the message length. Each one of these two approaches have their advantages and disadvantages. For simulation purposes, it has been decided to put a restriction on the length of the message that can be sent when erasure coding is enabled. The model requires that the message length be set to k times the block size. Future development to the model could remove this restriction.

The implementation of the erasure code will now be described.

The coding parameters (n,k) are sent as part of the ADDRESS_PDU. The field Total_Number_of_PDUs is loaded with the value k. An *erasure* field has been added to the ADDRESS_PDU format to receive the parameter n. The size of the field has been set to zero bits, although in reality this could not be done without multiplexing some fields or altering the packet structure. In any case, the new field introduces little amount of overhead when compared to the additional blocks necessary for encoding the source data.

The sender starts a transmission with the goal of sending up to n blocks of source data i.e. n DATA_PDUs. However, as soon as k DATA_PDUs have been sent, the sender should expect to receive the final ACK_PDUs from the receivers. If they are received, the sender aborts the transmission and sends an ADDRESS_PDU with an empty list of recipients to confirm having received the final ACK_PDUs. If they are not received, the sender keeps sending one more block of source data (DATA_PDU) at a time i.e. block number $k+1$, block number $k+2$, ..., block number n. After the transmission of any one of these blocks, the sender should check if it has received the final ACK_PDU from all of the receivers. If the final ACK_PDUs have been received, the sender should abort the transmission and send its final ADDRESS_PDU.

Assuming that the sender sent one full message (n blocks) but did not receive the final ACK_PDUs, the sender will enter the re-transmission phase. Which DATA_PDU should it be sending? There are several options:

1. It could retransmit the missing blocks in numerical order as is currently specified in the ACP without FEC; or,

2. It could try to be smart and figure out from the list of ACK_PDUs received which DATA_PDU (block of source data) is the most wanted and start by sending that particular DATA_PDU first; or,
3. It could enter a more complex scheme that remains to be specified.

The model currently implements the first option.

The procedure for the receivers is simple. The receivers know from the ADDRESS_PDU that they need to receive *k* blocks (DATA_PDUs) out of *n* to recover an error-free message. Therefore, they can notify the sender by sending a final ACK_PDU as soon as they receive *k* error-free DATA_PDUs. They also need not send any ACK_PDU until they have detected the end of the first transmission. A variant to this scheme would be for the receiver to start sending ACK_PDUs as soon as it knows it will never get *k* error-free DATA_PDUs from the first transmission. This additional complexity to the acknowledgment scheme has not been implemented.

3.3 Congestion Avoidance and Congestion Control

In P_MUL, ADDRESS_PDUs and ACK_PDUs are exchanged between source and destination hosts to effect which DATA_PDU packets need to be retransmitted to achieve reliable message transfer. To some extent, this mechanism regulates the flow of P_MUL packets, but only in so much as the amount of traffic is concerned. Even so, every destination node cares only about its own needs which can vary noticeably from one destination to the next because of potentially high variability in message reception quality. Like many other protocols, P_MUL does not truly regulate the transmission rate of the senders. In the ACP 142, the question of congestion avoidance and congestion control (CA/CC) is not addressed. On the other hand, the Internet-Draft on P_MUL briefly mentions the network congestion problem and specifies one way of dealing with ICMP messages indicating network congestion. There are underlying assumptions that are not explicitly declared in the document.

CA/CC becomes an issue when some links of the network are moderately to heavily loaded. In heterogeneous network environments, this situation can occur rapidly and quite frequently. However, since the choice of a particular CA/CC scheme over another can greatly influence the overall performance of any application it has been decided that the first release of the P_MUL model would not include any CA/CC element. For the simulation work presented herein, the transmitting nodes have the static capability to wait a certain amount of time (PDU_DELAY) between the sending of successive PDUs but none of them is actually controlled by ICMP messages or other feedback control signals. The study is oriented towards the functioning of P_MUL itself, in a congestion-free environment.

4.0 Simulation Information

The P_MUL model can primarily be configured by setting attributes of various elements or objects of the model. These attributes provide a means of controlling the behaviour of the model and many become important simulation parameters for assessing the protocol's performance. In this report, no attempt has been made to differentiate between model attributes and simulation parameters. In general, most simulation parameters mentioned in the report are indeed real attributes of the model, but not always. Some parameters may encompass a set of attributes and therefore hide from the reader some level of complexity that the authors did not judge appropriate to discuss in the space available for this document.

If model attributes and simulation parameters are useful *input* elements defining the operating conditions of the model, output statistics and performance metrics are necessary *output* elements for analysing the simulation results. The model has numerous predefined scalars and vectors to obtain measures of the protocol's performance or to make observations concerning its behaviour. In general, no attempt has been made to differentiate between the raw collection capabilities of the model and the performance metrics defined herein.

The simulation parameters and performance metrics are defined in Sections 4.1 and 4.2 respectively.

4.1 Simulation Parameters

For convenience, the simulation parameters have been separated into three main functional groups. The first group, introduced in Section 4.1.1, defines simulation parameters that are similar or closely related to the real parameters specified in the P_MUL protocol. The second group, presented in Section 4.1.2, describes the error-correction parameters for both P_MUL and the subnetworks. The third group, presented in Section 4.1.3, defines parameters surrounding P_MUL, i.e. parameters from other layers (transport, network, subnetwork) specifying an environment in which P_MUL can operate and be tested under well defined conditions.

4.1.1 P_MUL Parameters

The following arbitrary naming convention has been adopted to identify "formal" parameters of the P_MUL model: an all upper-case name designates a parameter defined in Annex A.1 of the ACP 142.

The P_MUL protocol model includes no less than 17 parameters whereas Annex A.1 of the ACP 142 only defines 8 parameters for P_MUL. Six of these parameters have been included in the model and their names preserved according to the convention mentioned above. Several procedures described in the ACP require functions that are subject to one

or more parameters not formally defined in Annex A.1. The approach used in such cases has been to create parameter names that closely resemble the function specified in the ACP. For instance, in Paragraph 426.a of the ACP, a statement such as "The ACK_PDU Timer shall be initialised for every ACK_PDU transmitted by a receiving node..." clearly identifies the need of providing the "ACK_PDU Timer" function with a timer duration value. In the model, this timer duration value is provided as part of a parameter named "ACK_PDU_Time". Note that the parameter name resembles the function name but since it is not part of the parameters formally defined in the ACP, its name has not been fully capitalised.

The main parameters used in the modelling of the P_MUL protocol itself are listed in Table 1. The parameters are grouped according to two main divisions: those parameters associated with, or being controlled by, the message transmitting nodes or "T-nodes"; and those being part of the message receiving nodes or "R-nodes". Each one of those two groups is further divided into three functional groups:

1. the *Core* functions, which mostly regroup parameters that define the basic functions of the protocol;
2. the *Dynamic Group Allocation* functions, which consist of parameters controlling the dynamic installation of multicast groups; and,
3. the *EMCON* functions, for EMCON or post-EMCON operating modes.

With regard to the EMCON group, it is worth mentioning that the ACP 142 identifies two operating conditions, EMCON and non-EMCON, and identifies three operating modes: EMCON mode, non-EMCON mode and "*receiving nodes in the non-EMCON mode having previously been in the EMCON mode*". In this report, the term *Post-EMCON* will be used to refer to the operating mode of a node that has left EMCON conditions and starts operating under non-EMCON conditions.

Parameters for the T-Node and R-Node groups are defined in Sections 4.1.1.1. and 4.1.1.2 respectively.

Table 1. Main simulation parameters of P_MUL model

P_MUL Parameter Group		Parameter	Unit
P_MUL (T-Nodes)	Core	ACK_Re-Transmission_Min	second
		ACK_Re-Transmission_Inc	second
		ACK_Re-Transmission_Max	second
		Max_Connection_Time_Out	second
		MPDU_SIZE	octet
		PDU_Delay	second
	Dynamic Group Allocation	ANNOUNCE_DELAY	second
		ANNOUNCE_ct	retransmission event
		WAIT_FOR_REJECT_TIME	second
	EMCON	EMCON_RTI	second
		EMCON_RTC	retransmission event
P_MUL (R-Nodes)	Core	ACK_PDU_Jitter	second
		Delete_DATA_PDUS_Time	second
		Min_Scan_Time	second
		M_Missing_DATA_PDUS	PDU
	Post-EMCON	ACK_PDU_Time	second

4.1.1.1 P_MUL T-Node Parameter Definition

A description of the simulation parameters defined for the T-Nodes follows.

ACK_Re-Transmission_Min

The ACK_Re-Transmission_Min parameter is used to initialised the ACK Re-Transmission Timer mentioned in Section 405 of the ACP 142. Its value sets the initial delay between retransmissions of DATA_PDUs that have not yet been acknowledged if there are one or more nodes that are not in EMCON. According to the ACP 142, the value is derived from the maximum round trip delay plus a safeguard.

ACK_Re-Transmission_Inc

This parameter holds the amount by which the ACK Re-Transmission Timer delay is increased each time the sender retransmits the same successive DATA_PDUs. Note that the retransmission delay cannot increase beyond ACK_Re-Transmission_Max.

ACK_Re-Transmission_Max

This parameter holds the maximum delay between retransmissions of the same successive DATA_PDUs if there are one or more nodes that are not in EMCON.

Note: The ACK_Re-Transmission_Inc and ACK_Re-Transmission_Max parameters are not mentioned in the ACP 142 but have been included in the model to investigate the performance of a retransmission scheme devised by the CSNI project members. The ACK_Re-Transmission Timer is first initialised with the minimum value ACK_Re-Transmission_Min specified in the ACP 142 then progressively incremented for each successive retransmission by an amount, ACK_Re-Transmission_Inc, up to a maximum ACK_Re-Transmission_Max. This scheme is also briefly described in Section 5.1.3.1 of the Internet-Draft for P_MUL. In the draft, Riechmann mentions that experiments proved that the duration of the timer should not be fixed. It is likely that drafters of the ACP 142 looked at this functionality as being an implementation issue rather than a specification issue.

Max_Connection_Time_Out

The P_MUL protocol relies on a network management function to terminate transmissions that have failed to close due to a node that is not responding. In the model, this function is replaced by the timeout timer. The Max_Connection_Time_Out parameter holds the maximum amount of time for which a message entry can exist. Once this interval has expired the simulated message is considered to have timed out and all remaining message information for the transmission is deleted.

MPDU_SIZE

The maximum size of any protocol data unit (PDU) created by P_MUL and sent to the transport layer for transmission over the multicast network.

PDU_Delay

This parameter holds the minimum delay between the transmission of consecutive PDUs for a message.

Note: The PDU_Delay parameter is not mentioned in the ACP 142. It is defined in Appendix A.2 and mentioned in Section 5.1 of the Internet-Draft for P_MUL. This parameter is to provide a means of dynamically controlling the transmission speed of a T-node while a message is being sent. It is one element of a congestion avoidance and congestion control (CA/CC) scheme devised by the CSNI project members. To avoid network congestion, the transmitting node has to wait a certain amount of time (PDU_Delay) before sending the next PDU. Each transmitting node must be prepared to receive ICMP messages indicating network congestion. As soon as a congestion message is received, the transmission of the

following PDUs are slowed down by increasing the actual value of PDU_Delay. Conversely, the value of PDU_Delay will be decreased as soon as for a certain amount of time no congestion message is received. It is likely that drafters of the ACP 142 looked at this functionality as being an implementation issue rather than a specification issue strictly related to the P_MUL protocol. This parameter (but no other elements of the CA/CC scheme) has been included in the model to investigate the effect of manually changing the transmission speed of a sending T-node.

ANNOUNCE_DELAY

This parameter holds the delay between successive retransmissions of ANNOUNCE_PDUs. In the event that the ANNOUNCE_ct is set to zero, it also specifies the time between sending an ANNOUNCE_PDU and the first affiliated ADDRESS_PDU.

ANNOUNCE_ct

This parameter holds the number of times that the transmitter will retransmit ANNOUNCE_PDUs.

WAIT_FOR_REJECT_TIME

This parameter holds the time that a transmitter who is negotiating a multicast address will wait to receive REJECT_PDUs before accepting the address and sending an affiliated ANNOUNCE_PDU.

EMCON_RTI

This parameter holds the delay between successive retransmissions of the entire message when all destinations are in EMCON mode.

EMCON_RTC

This parameter holds the maximum number of times that the sender will retransmit the entire message when all destinations are in EMCON mode.

4.1.1.2 P_MUL R-Node Parameter Definition

A description of the simulation parameters defined for the R-Nodes follows.

ACK_PDU_Jitter

Section 416.b of the ACP 142 mentions that to avoid the problem of ACK_PDU implosion at the message transmitting site, in the event that the receiving node has several ACK_PDUs to transmit each transmission of an ACK_PDU should be delayed by a random time value. The ACK_PDU_Jitter parameter is used to the establish a random delay bound to the predefined time interval of 0 to ACK_PDU_Jitter seconds.

Delete_DATA_PDUS_Time

The Delete_DATA_PDUS_Time parameter is used to initialise the Delete DATA_PDUS Timer mentioned in Section 413 of the ACP 142. This

parameter holds the time that a receiver will keep DATA_PDUs for which no ADDRESS_PDU has been received before discarding them. The ACP 142 does not suggest any default initialisation value.

Min_Scan_Time

This parameter is mentioned neither in the ACP 142 nor in the Internet-Draft on P_MUL. It has been used in the CSNI implementation of the protocol to hold the maximum amount of time for which a receiver will wait for a new PDU to be received for a particular message before sending a new ACK_PDU for that message.

M_Missing_DATA_PDUS

The M_Missing_DATA_PDUS parameter is mentioned in Section 421 of the ACP 142. This parameter holds the number of missing DATA_PDUs which will cause the receiver to send an ACK_PDU. Note that this value is also used as the maximum number of PDUs that may be listed in one ACK_PDU.

ACK_PDU_Time

The ACK_PDU_Time parameter is used to initialised the ACK_PDU Timer mentioned in Section 426 of the ACP 142. The timer controls the delay between the transmission of successive ACK_PDUs. The ACP 142 mentions that the ACK_PDU Timer should be initialised for every ACK_PDU transmitted by a receiving node in the non-EMCON mode, *having previously been in the EMCON mode*, and that the duration of the timer should be derived from the maximum round trip delay plus a safe-guard.

4.1.2 FEC Parameters

This section describes the forward-error-correction parameters used by P_MUL and the subnetwork links. The parameters most important for the simulation are listed in Table 2 and described thereafter.

Table 2. Parameters related to FEC

FEC Parameter Group		Parameter	Unit
FEC	P_MUL	Erasure Code (n,k) Erasure Block_Size	- octet
	Subnets	Transit ecc_abs_link (threshold) Transit ecc_rel_link (threshold) Destination ecc_abs_link (threshold) Destination ecc_rel_link (threshold)	bits percent bits percent

Erasure Code (*n*,*k*)

This parameter pair defines the number of source data blocks, *k*, that are used by the erasure coder at the sender to produce *n* blocks of encoded data.

Erasure Block_Size

This parameter defines the size of the erasure coding data blocks. A data block must fit into a single DATA_PDU and only one block is allowed per DATA_PDU. The model currently requires that the Message Length be set to *k* times the Block_Size.

Transit ecc_abs_link, Transit ecc_rel_link

This parameter pair defines the maximum number of correctable errors in a packet received by the data link layer over a point-to-point link. The number of acceptable errors is calculated as being equal to:
DLL packet size × Transit ecc_rel_link/100.0 + Transit ecc_abs_link
A packet containing more than the acceptable number of errors is rejected by the data link layer.

Destination ecc_abs_link, Destination ecc_rel_link

This parameter pair defines the maximum number of correctable errors in a packet received by the data link layer over a broadcast link. The number of acceptable errors is calculated as being equal to:
DLL packet size × Destination ecc_rel_link/100.0 + Destination ecc_abs_link
A packet containing more than the acceptable number of errors is rejected by the data link layer.

4.1.3 Stimuli and Environment

This section describes the simulation parameters associated with the controlled environment in which P_MUL is being studied. These parameters are composed of four main groups, which will be presented as follows:

- Network Topology parameters (Section 4.1.3.1);
- Source Subnet parameters (Section 4.1.3.2);
- Transit Subnet parameters (Section 4.1.3.3); and,
- Destination Subnet parameters (Section 4.1.3.4).

4.1.3.1 Network Topology Parameter Definition

A wide range of simulation topologies can be built around the generic P_MUL node models developed by CRC. A thorough description of these models is beyond the scope of this document. For the purpose of this study, the simulation topologies will consist of only three elements:

1. a source subnetwork, where a T-node host will issue one or more messages for distribution to some multicast group(s) located outside of his own subnetwork;
2. one or more transit subnetworks, containing no hosts affiliated to the multicast group(s), but participating in the message transfer by permitting the multicast traffic to reach its destinations; and,
3. one or more destination subnetworks, where R-node hosts will be waiting to receive the message(s) while operating in EMCON or non-EMCON conditions.

The characteristics and simulation parameters associated with these elements are described in Sections 4.1.3.2, 4.1.3.3 and 4.1.3.4. An example of a simple topology where a T-node host multicasts information towards several R-nodes located in four different subnetworks is shown in Figure 3.

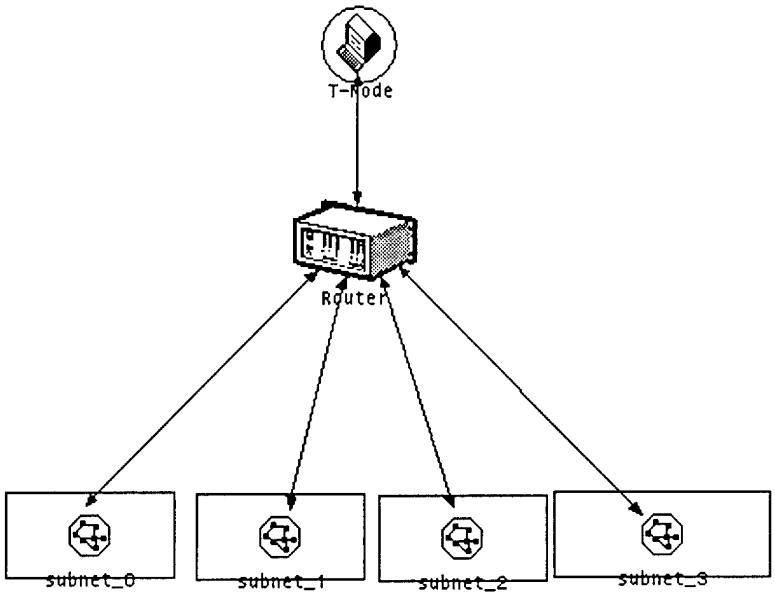


FIGURE 3. Example of a simulation topology used in this study

Creation of a topology is done via the OPNET Network Editor, a Graphical User Interface (GUI) allowing easy manipulation of modelling elements like those mentioned above. Specification of a topology is largely a matter of ensuring that hosts, nodes, links and other predefined simulated components are properly interconnected and assigned the correct identification numbers, such as host-network IP addresses, link port numbers, etc. The parameters most important for the simulation are listed in Table 3 and described thereafter.

Table 3. Parameters related to the network topology

Stimuli and Environment	Parameter	Units
Network Topology	Multicast Group (Address:Members) Group-size / Subnet-size	- members/nodes

Multicast Group

This parameter identifies the multicast group address and its members. The model supports up to 1000 unique multicast groups. Multicast group addresses must be a number between 1001 and 1999. The address 1000 is reserved for a multicast group consisting of all receivers on the network topology. Static multicast groups are specified by the user in a data file read by the simulation. Dynamic groups are chosen randomly by the model from the remaining available addresses of the pool. The members are identified by host and subnetwork identification numbers. These numbers must be assigned within the range 1 to 499 for the hosts and 500 to 999 for the subnetworks.

Group-Size / Subnet-Size

This parameter pair summarises the distribution of multicast members in the various subnets of the topology. Subnet-Size is the number of host nodes in a subnet whereas Group-Size is the number of members belonging to that subnet and to a given multicast group. If all the members of a group are located within the same subnet, which will be the case for most topologies studied in this report, then Group-Size represents the number of members in the multicast group.

Brackets around the parameters are used to indicate an enumeration of these values, one for each multicast group.

4.1.3.2 Source Subnet Parameter Definition

A typical source subnet is shown in Figure 4.

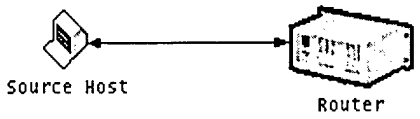


FIGURE 4. Example of a Source Subnetwork

It is composed of the following three elements:

1. A source host (T-Node), for sending messages over the network. Several messages can be sent concurrently if needed. Specifications for each message include the time at which the message will be sent, the length of the message, and a list of unicast or multicast group addresses (recipients) of the message.
2. A point-to-point link, for connecting the source host to a router. In a real tactical deployment, messages are likely to be generated from workstations on a LAN. In the model, this is represented by the source host and a high-speed, negligible delay, error-free point-to-point link for making connection to a backbone router.
3. A router, to interconnect the source subnetwork with the transit subnetwork.

The main source subnet parameters are listed in Table 4 and described thereafter.

Table 4. Parameters related to the source subnetworks

Stimuli and Environment	Parameter	Units
Source Subnets	Protocol (Multicast, Unicast or TCP)	-
	Message Length	octet
	default_latest_delivery_time	second
	Sender Transmission Rate	bits per second
	Msg Start Time: Expiry Time	second
	Multicast/Unicast Address	-
	Source Configuration File (none or default)	-
	Source Data Rate	bits per second
	Source Delay	second
	Source BER	errors per bit
	Source ptp_overhead_abs	bits
	Source ptp_overhead_rel	percent

Protocol

This term identifies the transmission protocol. Any message can be sent using one of three transport protocols: P_MUL Multicast, P_MUL Unicast, or TCP over IP. In the last two modes, messages are sent as individual unicast streams to each member of the destination list.

Message Length

This parameter holds the length of the message.

default_latest_delivery_time

This parameter holds the default value used to determine the message expiry time if a specific time is not provided with the message. The message expiry time will be set to the message start time plus default_latest_delivery_time.

Sender Transmission Rate

This parameter sets the nominal data transfer rate between P_MUL and the UDP protocol. Every source host (T-node) supports the transmission of several concurrent messages. Consequently, every message specification includes a Sender Transmission Rate parameter, which determines the bandwidth requested for the transmission of the message. Note that this rate is indirectly altered when a non-zero value is set for the PDU_Delay parameter (see Figure 5) Also, in the current CRC implementation of the model, a PDU_Delay value cannot be independently specified for each message. Instead, all messages at a T-node share a common PDU_Delay timer setting. This means that all outgoing messages are equally delayed by the same amount. Such configuration may not be desirable in an environment where congestion control is selectively quenching the sources by looking at the relative priority of the messages.

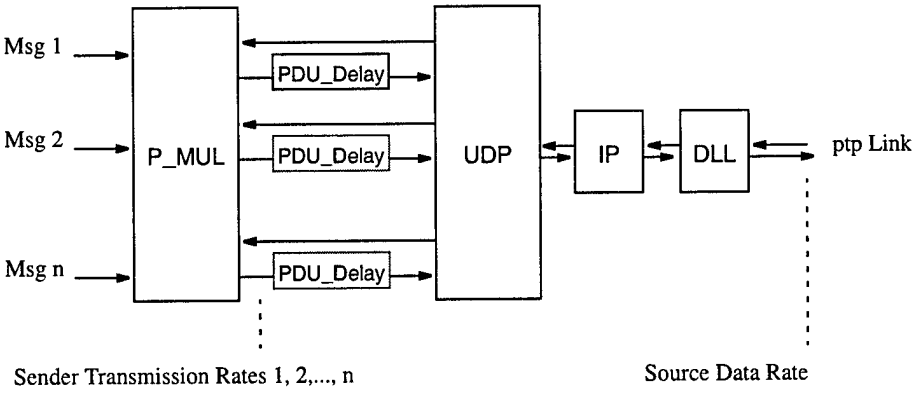


FIGURE 5. Parameters controlling the message transfer speed at a T-Node

Msg Start Time : Expiry Time

This parameter pair holds the start time and expiry time of the message as requested by the application. The start time is the time at which P_MUL begins the message transfer process. The expiry time is the time at which the message is considered outdated and can be discarded.

Multicast/Unicast Address

This parameter holds the destination address of the message to be sent.

Source Data Rate

This parameter sets the transmission rate on the point-to-point link connecting the source host (T-node) to the rest of the network (Figure 5). Unless otherwise indicated, the link is assumed to be high bandwidth so this parameter is set at 10 Mbps.

Source Delay

This parameter sets the propagation delay on the point-to-point link interconnecting the source host (T-node) to the backbone router. Unless otherwise indicated, the link is assumed to have negligible delay so this parameter is set at 0 seconds.

Source BER

This parameter sets the bit error rate on the point-to-point link interconnecting the source host (T-node) to the backbone router. Unless otherwise indicated, the link is assumed to be error-free so this parameter is set at 0 errors per bit.

Source ptp_overhead_abs

This parameter holds the static number of overhead bits added to the IP packets. This overhead is meant to represent the header information added by the link layer protocol before the packet is sent over the point-to-point link connecting the source host (T-node) to the rest of the network.

Source ptp_overhead_rel

This parameter holds the percentage of the IP packet size added as overhead to the IP packets. This overhead is meant to represent the redundant information added to the link service data unit (SDU) by the link layer protocol to achieve some degree of link reliability.The total amount of link overhead added to IP packets is calculated as:

IP packet size × ptp_overhead_rel/100.0 + ptp_overhead_abs

4.1.3.3 Transit Subnet Parameter Definition

A simple transit subnetwork is shown in Figure 6.

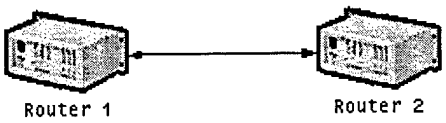


FIGURE 6. Example of a Transit Subnetwork

It is composed of a single point-to-point link interconnecting two backbone routers. Source and destination subnetworks can be attached to the end routers to create a complete network topology. The main transit subnet parameters are listed in Table 5 and described thereafter.

Table 5. Parameters related to the transit subnetworks

Stimuli and Environment	Parameter	Units
Transit Subnets	Transit Data Rate	bits per second
	Transit Delay	second
	Transit BER	errors per bit
	Transit ptp_overhead_abs	bits
	Transit ptp_overhead_rel	percent
	Transit Background Traffic Load	percent

Transit Data Rate

This parameter sets the transmission rate on the point-to-point link interconnecting the two backbone routers.

Transit Delay

This parameter sets the propagation delay on the point-to-point link interconnecting the two backbone routers.

Transit BER

This parameter sets the bit error rate on the point-to-point link interconnecting the two backbone routers. The model uses the uniform error distribution model provided by OPNET for the default point-to-point link model.

Transit ptp_overhead_abs

This parameter holds the static number of overhead bits added to the IP packets. This overhead is meant to represent the header information added

by the link layer protocol before the packet is sent over the point-to-point link interconnecting the two routers.

Transit ptp_overhead_rel

This parameter holds the percentage of the IP packet size added as overhead to the IP packets. This overhead is meant to represent the redundant information added to the link service data unit (SDU) by the link layer protocol to achieve some degree of link reliability.The total amount of link overhead added to IP packets is calculated as:

IP packet size × ptp_overhead_rel/100.0 + ptp_overhead_abs

Transit Background Traffic Load

This parameter holds the percentage of the link capacity allocated to background traffic generated on the transit subnetwork links to emulate the use of the subnetwork by other applications unrelated to P_MUL. The load is on average constant but can be made to follow various probability distributions. The number of parameters required to specify the load varies with the distribution and is not detailed herein.

4.1.3.4 Destination Subnet Parameter Definition

A typical destination subnetwork is shown is Figure 7.

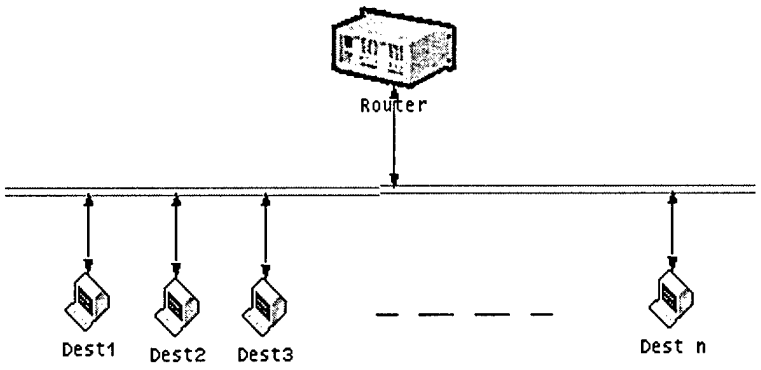


FIGURE 7. Example of a Destination Subnetwork

The destination subnetwork is composed of the following elements:

1. A router, to interconnect the destination hosts (R-nodes) to the transit subnetwork.

2. A bus link, emulating a broadcast medium such as an army-terrestrial or navy-fleet radio net or some other line-of-sight radio net. The model assumes negligible propagation delay between the hosts but will accept any level of transmission error specification between 0 and 1. All hosts on the bus are subjected to the same bit error rate but bit errors are independent from host to host. The errors are uniformly distributed in the packets exchanged over the bus.
3. One or more destination hosts (R-Nodes), for receiving messages from the network. Specification for each destination host include, whenever applicable, the time at which the host will enter and leave the EMCON mode.

The main destination subnet parameters are listed in Table 6 and described thereafter.

Table 6. Parameters related to the destination subnetworks

Stimuli and Environment	Parameter	Units
Destination Subnets	Destination Data Rate	bits per second
	Destination Delay	seconds
	Destination BER	errors per bit
	Destination bus_overhead_abs	bits
	Destination bus_overhead_rel	percent
	Destination Background Traffic Load	percent
	EMCON Period (Node, Start time, End time)	-, second, second

Destination Data Rate

This parameter sets the transmission rate on the bus link shared by the destination hosts and the backbone router.

Destination Delay

This parameter sets the propagation delay on the bus link shared by the destination hosts and the backbone router. The bus is meant to emulate a broadcast medium where propagation delays are negligible. The model does not currently support a non-zero value for this parameter.

Destination BER

This parameter sets the bit error rate on the bus link shared by the destination hosts and the backbone router. The model uses the uniform error distribution model provided by OPNET for the default bus link model.

Transit bus_overhead_abs

This parameter holds the static number of overhead bits added to the IP packets. This overhead is meant to represent the header information added by the link layer protocol before the packet is sent over the bus link shared by the destination hosts and the backbone router.

Transit bus_overhead_rel

This parameter holds the percentage of the IP packet size added as overhead to the IP packets. This overhead is meant to represent the redundant information added to the link service data unit (SDU) by the link layer protocol to achieve some degree of link reliability.The total amount of link overhead added to IP packets is calculated as:

IP packet size × bus_overhead_rel/100.0 + bus_overhead_abs

Destination Background Traffic Load

This parameter holds the percentage of the bus capacity allocated to background traffic generated on the destination subnetwork bus to emulate the use of the subnetwork by other applications unrelated to P_MUL. The load is on average constant but can be made to follow various probability distributions. The number of parameters required to specify the load varies with the distribution and is not detailed herein.

4.1.3.5 Other Simulation Parameters

udp.add_overhead, ip.add_overhead

These two parameters are software toggles used by the UDP and IP protocol models to include or exclude the UDP and IP overhead in the calculation of the UDP and IP packet sizes. If disabled, the UDP and IP headers are transmitted as normal packets but wondrously introduce 0 bits of overhead.

4.2 Performance Metrics

This section defines various terms and performance metrics needed for the simulation scenarios and the analysis presented in Section 5.0. P_MUL performance is assessed across three main axes:

1. response time metrics, to measure how fast messages are delivered by P_MUL;
2. transmission capacity metrics, to measure how much data is successfully transferred by P_MUL in a given time period; and,
3. overhead metrics, to determine how much of the network resources are utilised by P_MUL when delivering messages.

In this report, the term "successful", as in "successful receiver" for instance, is used to identify the condition where the sender of the message has received acknowledgement of the message. The context will usually indicate whether the success condition is for one or more receivers (R-nodes).

The metrics are presented in Sections 4.2.1 to 4.2.5, the closing section being reserved for some other selected metrics not belonging to any one of the three groups discussed above but providing useful information on the behaviour of the protocol.

4.2.1 Response Time

P_Delivery Time

The elapsed time measured at a T-node between the sending of a P_MUL message and the receipt of the final acknowledgement from all members affiliated to the multicast group. The elapsed time therefore includes the time for transmitting the message and, for all R-node receivers, the time to receive and send acknowledgement.

Normalised P_Delivery Time

The P_Delivery Time normalised to the time the message transfer would take to send at the nominal destination subnetwork capacity. The normalised P_Delivery Time is calculated as:

P_Delivery Time / (Message Length / Destination Data Rate)

Message Active Time

The elapsed time between the start and the end of the message transfer process. For messages successfully delivered to all destinations of the multicast group, the elapsed time usually corresponds to the P_Delivery Time metric. For messages where one or more receivers were not successful, this elapsed time includes a latency period ended either by the message expiry time, the Max_Connection_Time_Out set for the message (Section 4.1.3.2) or the end of the simulation run.

Time taken to receive message

The time taken for each R-node to receive the entire message. If a receiver failed to successfully receive the message, the time taken will be recorded as 0 seconds.

Avg time to receive message

The average time taken by the R-nodes to receive the entire message.

Ratio of last DATA_PDU received time to message received time

The ratio of the last DATA_PDU received time to the message received time for each receiver in a message.

Latest DATA_PDU received

The elapsed time between the message start time, as defined by the Msg Start Time parameter (Section 4.1.3.2), and the time at which the last

DATA_PDU was received among all the receivers who successfully received the message.

Time to Negotiate Message

The time taken by the T-node to negotiate a multicast address before sending the message.

4.2.2 Capacity

Throughput

The throughput achieved at a T-node is calculated as the Message Length in bits divided by the P_Delivery Time in seconds. For unsuccessful messages, the throughput is zero.

Effective throughput

The effective throughput achieved at a T-node is calculated as the Message Length in bits times the number of successful receivers all divided by the P_Delivery Time. In the ideal case, the effective throughput from one sender to n receivers should be n times the unicast rate.

4.2.3 Overhead

Link Utilization

The average fraction of the destination bandwidth used by P_MUL during the Message Active Time (Section 4.2.1). This metric is calculated by summing the time periods during which P_MUL PDUs are occupying the destination link and dividing by the value of Message Active Time.

Packet Ratio

The Packet Ratio achieved at a T-node is calculated as the ratio of the number of PDUs actually transmitted to the minimum number of PDUs that theoretically needed to be sent. In the ideal case where a message is fully acknowledged at a T-node by all the members of the multicast group after the sending of the first transmission, the Packet Ratio is 1.

4.2.4 Overall Performance

Message Transfer Efficiency (MTE)

This metric is the ratio of the throughput achieved at a T-node to the bandwidth consumed at the destination subnet during a message transfer:

$$MTE = \frac{\text{Throughput}}{\text{Nominal Link Capacity} \times \text{Link Utilization}} \quad (1)$$

The MTE can equivalently be expressed in term of the throughput definition given in Section 4.2.2; it is the ratio of the amount of information successfully transferred to the amount theoretically transferable during that same time period:

$$MTE = \frac{(\text{Message Length} / \text{P\_Delivery Time})}{\text{Nominal Link Capacity} \times \text{Link Utilization}} \quad (2)$$

4.2.5 Other

Number of addresses rejected

The number of dynamic addresses chosen for the message for which valid REJECT_PDUs were subsequently received. This statistic is only recorded if a dynamic multicast address was needed for the message.

Total number of acks received

The total number of ACK_PDUs received at a T-node.

Total number of packets received by receivers

The total number of DATA_PDUs, ADDRESS_PDUs, and DISCARD_PDUs received by the receivers of the message.

Total number of packets sent

The total number of packets sent by the T-node.

Total number of packets rejected by pmul

The total number of packets that were rejected by P_MUL because they contained more errors than the calculated acceptable threshold.

Total number of packets received in error by receivers

The total number of DATA_PDUs, ADDRESS_PDUs, and DISCARD_PDUs received by the receivers of the message which contained one or more errors.

Total number of errors received

The total number of errors received in DATA_PDUs, ADDRESS_PDUs, and DISCARD_PDUs transmitted for the message.

Avg errors per packet

The average number of errors received per DATA_PDU, ADDRESS_PDU, or DISCARD_PDU received for the message. The average is calculated by

dividing the total number of errors received by the total number of packets received.

Number of Retransmissions

The number of times the sender retransmits in full or in part the message. The count is incremented with the sending of every new ADDRESS_PDU. Therefore, if a single transmission is required to successfully deliver the message then the Number of Retransmissions is equal to one.

5.0 Simulation and Analysis

Because of the numerous problems encountered during the development phase of the P_MUL model, an exhaustive study of the protocol could not be performed within the time frame available for this project. Nevertheless, the study presented here meets many of its initial objectives, including:

1. To provide performance data such as messaging throughput, delivery time and overhead for messages being multicast over an unreliable broadcast network;
2. To provide dynamic group installation performance for groups of various sizes;
3. To determine the robustness of the protocol to bit errors;
4. To study the impact of erasure block coding for retransmission;
5. To determine the protocol's response to various changes in predefined parameter configuration values;
6. To verify that P_MUL can be used to transfer large files as mentioned in the Internet-Draft.

Detailed results of the study are presented into two parts (Sections 5.1 and 5.2). The main findings of the study are also summarised in Section 6.0.

In Section 5.1, a series of short simulation scenarios are proposed where the core parameters of P_MUL are varied over selected ranges of values. Snap-shots of the protocol's response are described and comments are given. The whole section uses a heuristic approach to provide a quick exploration into the P_MUL protocol core. These simple scenarios turned out to be quite useful in building up confidence in the model during its development phase and may help readers unfamiliar with the protocol to grasp some of the subtleties of P_MUL. Scenarios in Section 5.1 assumed that all communication links in the network are error-free.

The second section, Section 5.2, includes a more detailed analysis. Noise is injected into the communication channels and packets are rejected when randomly corrupted by bit errors. Some analytical expressions are also derived and presented to help clarify the results. A few important deficiencies of the P_MUL protocol are identified and these will likely command changes to the standard.

5.1 Reliable Network Service

This section is a collection of introductory scenarios. All communication links are error-free and messaging packets always reach their destination in the same order as they were transmitted by the sender.

These exploratory scenarios are derived from one common topology, which is shown in Figure 8.

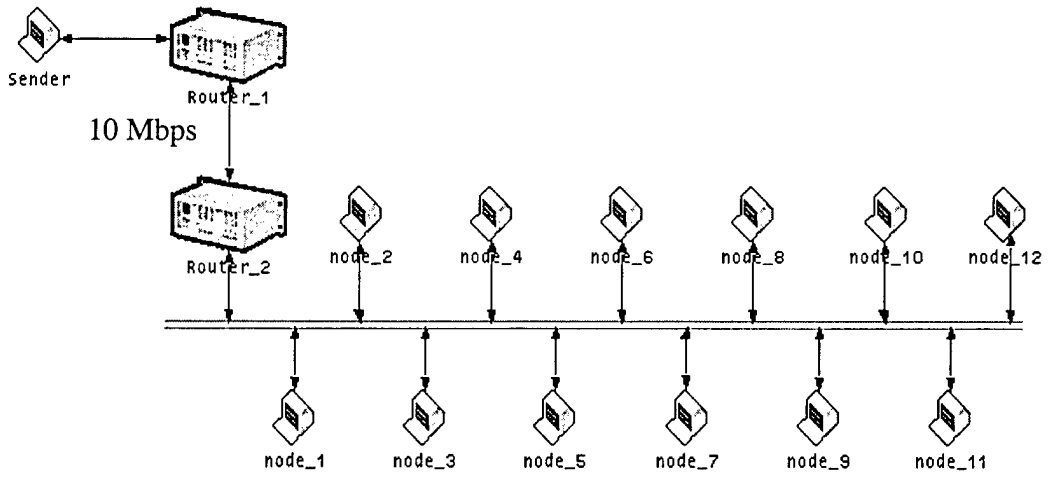


FIGURE 8. Simulation topology A

The topology is composed of one T-node (sender), one high-speed (10 Mbps) short delay (50 ms) point-to-point transit link between the source and destination routers and a 64 kbps destination broadcast network where resides a multicast group of 12 recipients (node_1 to node_12). It is assumed that all nodes are operating in the non-EMCON condition. The communication channels are error-free even though some arbitrary overhead has been allocated to simulate reliable links. The default overhead values as well as other environment parameters most important for the scenarios of this section (Section 5.1) are summarised in Table 7.

Table 7. Default parameter settings for simulation topology A

	Parameter	Default Values
Source Subnet	Message Length	30 ko
	Sender Transmission Rate	64 kbps
	Source Data Rate	64 kbps
	Source ptp_overhead_abs	64 bits
	Source ptp_overhead_rel	0%
Transit Subnet	Transit Data Rate	10 Mbps
	Transit Delay	.05 s
	Transit ptp_overhead_abs	64 bits
	Transit ptp_overhead_rel	7%
Destination Subnet	Destination Data Rate	64 kbps
	Destination bus_overhead_abs	96 bits
	Destination bus_overhead_rel	10%

5.1.1 No Re-Transmissions

One of the most striking characteristics of P_MUL is the fact that the protocol can be very "silent" when the settings and conditions are right. The first of two "*No Re-Transmissions*" scenarios falls into that category. It is assumed that a multicast group has been installed manually. The T-node is requested to deliver a 30 000 octet message. The maximum PDU size has been set to 1024 octets so that the message, including any protocol overhead, fits nicely into 30 DATA_PDUs. The ACK Re-transmission timer is set to an excessively large value to prevent any retransmission from occurring. After issuing an ADDRESS_PDU, to notify the receivers of the upcoming message, the sender transmits its 30 DATA_PDUs one after another, without any interruption. It does not matter whether the message is 30, 300, 3000 or 3 billion DATA_PDUs long, the same scenario is being played. The sender continuously tansmits its message while the receivers quietly listen and collect the message, packet by packet, until the last DATA_PDU is received. Only then does each receiver break the radio silence and transmit one acknowledgement PDU to notify the sender that the message has been received in its entirety. These single ACK_PDUs make the sender happy because they mean that there is no need to retransmit the message, in whole or in part, which, in this scenario, is a blessing because the sender initiated the transmission without provision for any retransmission. After receiving the ACK_PDUs, the sender will issue a final ADDRESS_PDU to notify the receivers that the message transfer has been successfully acknowledged and is completed. In summary, the traffic flow consists of only 44 PDUs:

- 1 ADDRESS_PDU to notify the receivers of the upcoming message;
- 30 DATA_PDUs to transfer the message;
- 12 ACK_PDUs to acknowledge the receipt of the message by the receivers; and,
- 1 final ADDRESS_PDU to notify the receivers that the message transfer has been successfully acknowledged and is completed.

Obviously, the scenario can only work because the network is reliable. A congested network where packets are dropped could not possibly be without retransmissions. In fact, if the network does benefit from this long period where receivers are muted and consequently help keep the overall traffic load of the network to a minimum, this lengthy absence of any feedback to the sender can affect the delivery performance of the message over noisy channels as it will be shown later in other scenarios.

There would not be much more to say about a *No Re-Transmission* scenario if it were not for the way that ACK_PDUs are transmitted by the receivers. To avoid the problem of ACK_PDU implosion at the sender, the transmission of every ACK_PDU is delayed locally by a random amount of time. The delay is uniformly distributed within a predefined range of 0 to ACK_PDU_Jitter seconds. To illustrate the use of the ACK_PDU_Jitter parameter, three maximum jitter settings are chosen for this scenario: 0.01, 2.5 and 5.0 seconds.

The parameters most relevant to the simulation scenario described so far are summarised in Table 8 with selected results being plotted in Figure 9. The transmission rate at the

sender is normalised to the nominal subnetwork bandwidth (64 kbps) and was sampled over a range of 10 to 100 kbps, in step of 2 kbps, necessitating 46 simulation runs to obtain every trace shown in the graphs. The procedure was executed three times, once for every ACK_PDU_Jitter setting.

Table 8. Main parameter settings for scenario A010[1]

	Parameter	Default Value	Scenario A010
P_MUL Core	ACK_Re-Transmission_Min MPDU_SIZE PDU_Delay	 1024 o 0.0 s	infinity s
	ACK_PDU_Jitter		one of {0.01, 2.5, 5.0} s
Source Subnet	Sender Transmission Rate	64 kbps	*variable*

1. Values in italic overwrite default values shown in Table 7.

For an ACK_PDU_Jitter setting of 0.01 second, which represents a simulated case where the jitter delay is negligible, the P_Delivery Time is nearly equal to the Average Receive Time of the message by all receivers. This is due to the small ACK_PDU size and the short transmission delay (50 ms) over the transit link. This result is not shown in Figure 9 since the two curves would nearly overlap.

For ACK_PDU_Jitter settings of 2.5 and 5.0 seconds, the message delivery is delayed by an amount close to the maximum jitter value and not, like some could perhaps be tempted to assume, by the mean value of the jitter window. In fact, the message delivery time could be delayed by an amount greater than the jitter window if two or more receivers randomly pick a jitter delay near the maximum edge of the jitter window, or when the group membership is too large for the selected window jitter size. In both of these cases ACK_PDUs may be queued on the link going to the sender.

In an error-free environment, large ACK jitter would only delay the acknowledgement of the message by the receivers, not the time for the latter to receive the message. Note that this is not necessarily true when there are errors on the link.

In both plots of Figure 9, the curves flatten out when the subnetwork reaches saturation around 0.87, i.e., the message delivery and throughput are nearly constant when the sending rate exceeds the subnet nominal transmission capacity. It is clear that too large a jitter window can significantly reduce P_MUL's performance. For the remainder of this study, an ACK_PDU_Jitter setting of 1.0 second will be used as the default jitter setting. This value is rather conservative because an empty ACK_PDU is only 24 octets in size. When accounting for the UDP, IP and DLL overhead, about one hundred of them can be transmitted in less than 1 second if the transmission rate is 64 kbps.

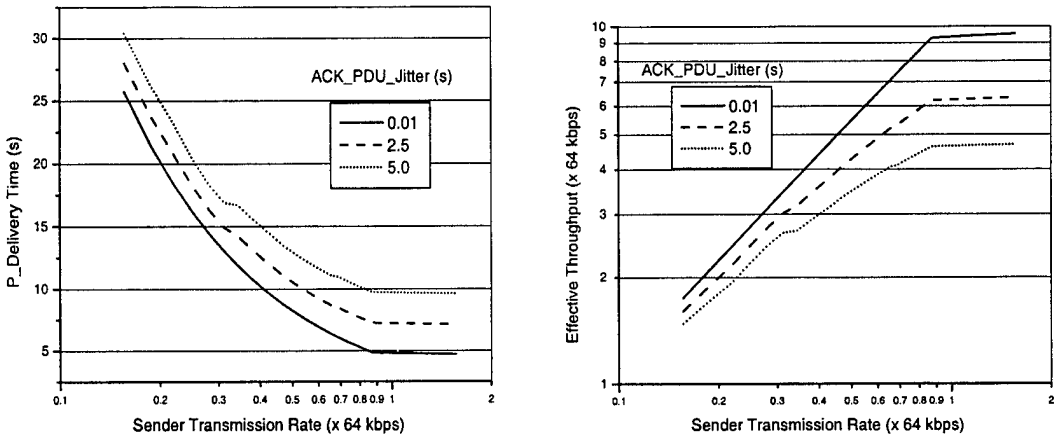


FIGURE 9. Effect of jitter window on performance

The second and last scenario under the "No Re-Transmission" heading also has to do with how ACK_PDUs are transmitted at the receivers. In this scenario, A020, a combination of parameter settings show that it is possible to generate many more ACK_PDUs than needed during the course of a single unrepeated message transmission. The parameters most relevant to this scenario are listed in Table 9 with selected results being plotted in Figure 10. The bit transmission rate at the sender is kept constant at 64 kbps but the transmission of every packet is delayed according to the setting of the PDU_Delay parameter. The latter was sampled over a range of 0 to 100 seconds, in steps of 2 seconds, necessitating 51 simulation runs to obtain every trace shown in the graphs. The procedure was executed four times, once for every Min_Scan_Time setting.

Table 9. Main parameter settings for scenario A020

	Parameter	Default Value	Scenario A020
P_MUL Core	ACK_Re-Transmission_Min		infinity s
	MPDU_SIZE	1024 o	
	PDU_Delay		variable
	ACK_PDU_Jitter	1.0 s	
	Min_Scan_Time		one of {15,30,45,60} s
Source Subnet	Sender Transmission Rate	64 kbps	

As mentioned in Section 4.1.1, the Scan timer is not a parameter defined in the ACP 142. It has been used in the CSNI implementation of the protocol as a means of re-initiating the sending of an ACK_PDU in the case where a receiver detected an incomplete message, perhaps due to the lost of an ADDRESS_PDU for example. As the results show in Figure 10, the Scan timer has negligible effect on the message delivery time. In this sce-

nario, it only forces more ACK_PDUs to be received at the sender. The number of acknowledgements received is granular, i.e., their number steps up by a well defined amount every time the PDU_Delay setting exceeds a multiple of the Min_Scan_Time setting. Note that in this scenario, even though the sender receives many acknowledgments (lists of DATA_PDUs missing at a receive site), it never retransmits any DATA_PDUs because all of the acknowledgements are received during the first (initial) transmission of the message.

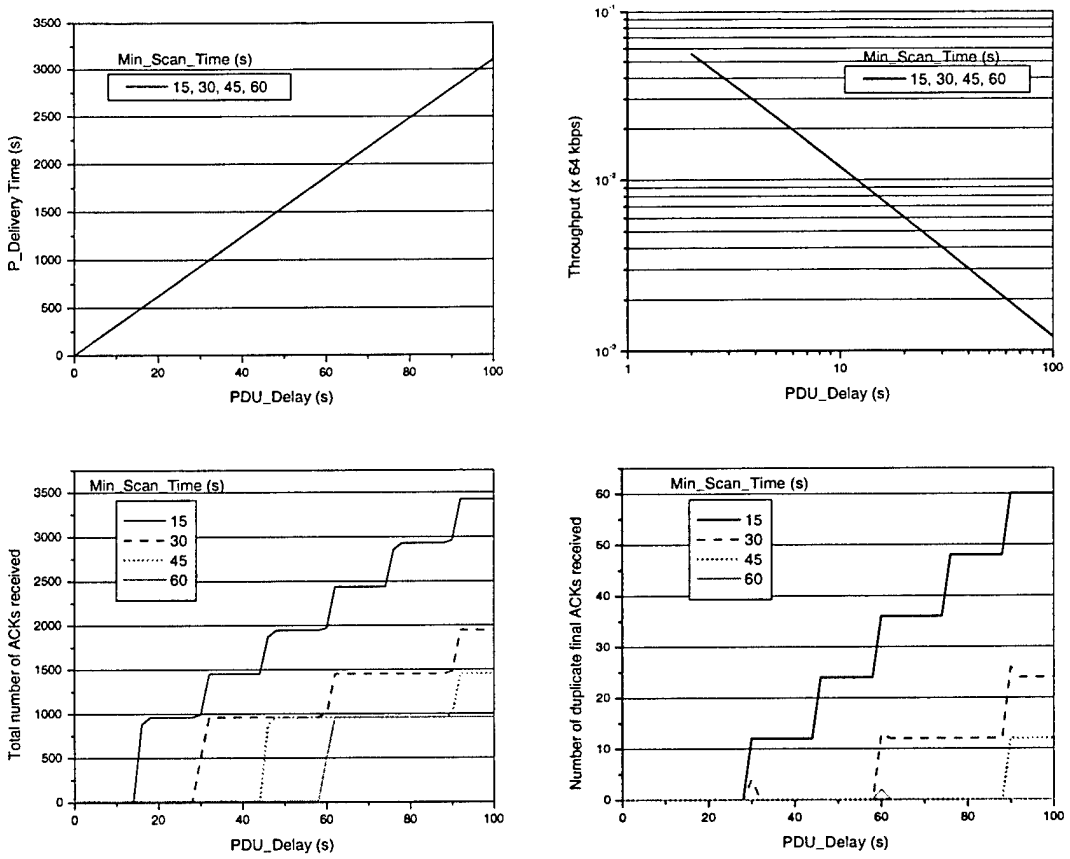


FIGURE 10. Use of the Scan timer to initiate the sending of ACK_PDUs

5.1.2 Constant Re-Transmissions Scheme

In previous scenarios the ACK Re-Transmission timer was set to infinity. Only one acknowledgement received from each node is sufficient to confirm the correct receipt of a message. There was no need to retransmit the message and therefore no need to initialise the timer. Such a setting is not realistic since most networks are likely to experience reliability or congestion problems at one time or another. The ACP 142 specifies that the ACK Re-Transmission timer should be initialised with a value "*derived from the maximum round trip delay plus a safeguard*". It is not clear what assumption the standard makes about the probe used to estimate the maximum *round trip* delay. If no probe is used then

the maximum round trip delay is simply calculated as being twice the sum of the propagation delays along the path connecting the sender to the farthest (slowest) destination. Otherwise, the maximum delay is the time taken by "something" to travel from the sender to the farthest (slowest) destination and back to the sender[1]. Presumably, the intent here is to specify a delay that helps coordinate the timing associated with the handshake taking place between the sender and the receivers during a message transfer. The delay should give the receivers sufficient time to respond to the sender when a need for the retransmission of some missing PDUs arise. For that reason, the probe may well consist of:

1. a PDU of maximum size; or perhaps,
2. the whole message itself[2].

At first sight, the ACK_Re-Transmission interval should preferably not be dependent of the message length and a probe based on the maximum PDU size would appear to make more sense. Nevertheless, these two possibilities have been retained and will often be analysed together in many of the scenarios to be presented in this study.

The ACK Re-transmission timer has one important intrinsic characteristic. Although it has not been specifically stated in the ACP, the timer is aperiodic. That is, the timer is re-initialised after every complete or partial message transmission as opposed to being initialised once and left alone while it generates periodic retransmission events.

Another issue worth mentioning about the ACK Re-transmission timer is the option for two re-initialisation modes:

1. The ACP 142 specifies that the timer is initialised and reinitialised with the maximum round trip delay plus a safeguard;
2. The Internet-Draft adds a progressive re-transmission scheme where every new re-initialisation value is increased by a certain amount or factor.

The issue is in part implementation dependent but can certainly impact the performance of the protocol. By default, the constant re-transmission scheme of the ACP will be used throughout this study, except for scenario A040 in Section 5.1.3 where the usefulness of the progressive scheme will be investigated.

The parameters most relevant to the current scenario, A030, are listed in Table 10. The bit transmission rate at the sender is kept constant at 10 Mbps, an excessively high value considering that the nominal destination subnet capacity is only 64 kbps. This causes one or more retransmissions to occur before the sender receives the receivers' acknowledgements to the first message transmission. As in the previous scenarios, the transmission of every packet is delayed according to the setting of the PDU_Delay parame-

---

1. An example might be a DATA_PDU going out and an ACK_PDU coming back.
2. Actually, the time taken to send the entire message in the sender-receivers direction and a PDU of maximum size in the return direction.

ter. The latter was sampled over a range of 0 to 0.2 seconds, in steps of 0.002 seconds, necessitating 101 simulation runs. An ACK Re-transmission value of 0.5 seconds was allocated to the timer duration based on the following maximum round trip delay estimate:

$$D_1 \approx 2 \times (\text{Transmission Delay} + \text{Path Delay}) + \text{Safeguard} \quad (3)$$

where

$$\text{Transmission Delay} = \frac{8 \times (\text{MPDU\_SIZE} + \text{UDP:IP:DLL Overhead})}{\text{Nominal Subnet Capacity}} \quad (4)$$

The Nominal Subnet Capacity is that of the slowest link in the network, i.e. the 64 kbps destination network. The Path Delay is the sum of the propagation delays between the source and the destination that is the farthest distant in time. The one-way transmission and path delays are about 0.13 and 0.05 seconds respectively whereas the safeguard accounts for about 27% of the overall round trip delay estimate.

Table 10. Main parameter settings for scenario A030[1]

	Parameter	Default Value	Scenario A030
P_MUL Core	ACK_Re-Transmission_Min		0.5 s
	MPDU_SIZE	1024 o	
	PDU_Delay		variable
	ACK_PDU_Jitter	1.0 s	
	Min_Scan_Time	60 s	
Source Subnet	Sender Transmission Rate	64 kbps	*10 Mbps*

1. Values in italic overwrite default values shown in Table 7.

Selected results are plotted in Figure 11. The best throughput is achieved when the PDU_Delay is set to zero. At best, the message is delivered in 5.54 seconds, during which about 1 second is being spent on randomly delaying the acknowledgement PDUs before their transmission. Unfortunately, while waiting for the acknowledgements of the first message transmission, the sender had time to retransmit the same message 11 times. These numerous additional transmissions loaded the network for more than 44.46 seconds (Message Active Time = 50 s) after the message had been fully acknowledged. This redundant traffic consisted of about 200 ACK_PDUs and 300 DATA_PDUs. Clearly, the retransmission timer selected was too short to accommodate this heterogeneous network configuration. Setting the timer to the maximum round trip delay of a maximum-size PDU ensures an excellent message delivery time but can also possibly harm the network through the presence of several unnecessary retransmission. The effect of increasing the timer duration will be analysed shortly, but first a word must be said about the PDU_Delay parameter.

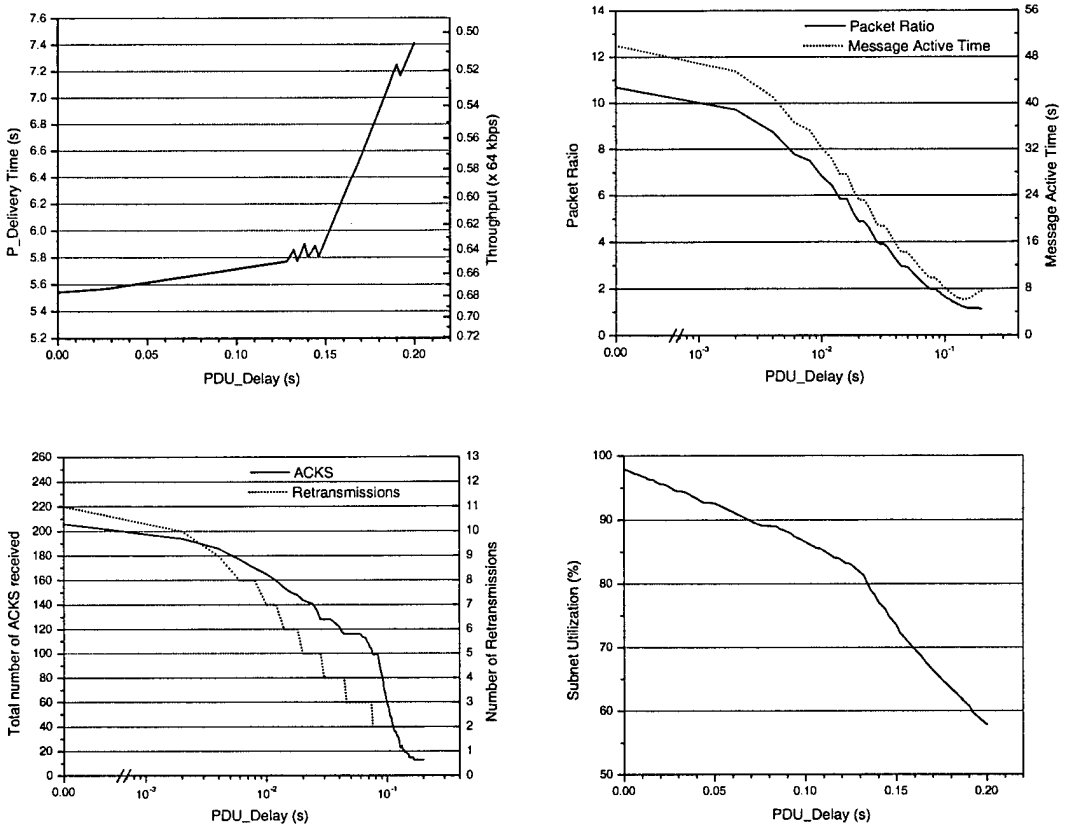


FIGURE 11. Use of the PDU_Delay parameter to prevent heavy number of retransmissions

In a tactical communications environment, everyone is aware of the importance of not excessively driving a narrow band network with a high speed source. In a sense, scenario A030 is representative of a system mis-configuration or the presence in the network of a non well-behaved application. Networks can rely on their congestion avoidance and congestion control (CA/CC) mechanisms to prevent such a source from overloading the network. Figure 11 shows that the PDU_Delay parameter plays this role and has a strong impact on the frequency of the retransmissions triggered at the sender and the number of ACK_PDUs generated by the receivers. A low PDU_Delay value produces many more retransmissions and acknowledgements than needed. Some relief is gradually introduced as the delay is increased.

The panel working on the ACP 142 standard decided not to include the PDU_Delay parameter in the P_MUL draft presumably because its members felt, like so many standardisation committees have come to realise for other protocols in the past, that CA/CC is an issue that does not belong to the specification scope of the protocol at stake. In the Internet-Draft of P_MUL, Riechmann takes a different approach and proposes a scheme where Internet Control Message Protocol (ICMP) messages are used to adjust the value of the

PDU_Delay parameter whenever congestion is detected in the network. The scheme is simple, but the whole issue of CA/CC is a complex one. It always involves many elements of a system. Is the scheme able to avoid any conflict with other CA/CC mechanisms which could result in improper feedback signal to control the traffic flow? The authors feel it is safer to leave the issue out of the standard as it has been done in the ACP or else it would be important to specify the scheme in more details.

In every scenario presented in this study, the PDU_Delay parameter is set once at the start of a simulation and kept constant for the duration of the run. There is no feedback signal derived from the traffic load to dynamically adjust the delay value as the simulation progresses.

So, if the dynamic control of the PDU_Delay parameter is suppressed, how does P_MUL cope with message transmission rates widely different than those supported by some destination subnetworks? Without deviating too much from what has been proposed in the ACP and the Internet-Draft, there are at least three non-exclusive venues:

1. The PDU_Delay parameter can be set with a non-zero static value;
2. The ACK Re-transmission timer can be set with a value other than the maximum round trip delay; and,
3. The ACK Re-transmission timer can be progressively readjusted by a static value or ratio prior to every new retransmission.

As previously mentioned, the third option is presented in scenario A040 in Section 5.1.3. The remainder of this section will now discuss the second option.

In scenario A035, the ACK Re-transmission timer is sampled over the range 0.5 to 10 in steps of 0.1 second. The same fast transmission rate of 10 Mbps is used at the sending node whereas a delay of zero seconds is set for the PDU_Delay timer. The parameter settings for the scenario are shown in Table 11. The simulation necessitated 96 runs. The results are presented in Figure 12.

Table 11. Main parameter settings for scenario A035[1]

	Parameter	Default Value	Scenario A035
P_MUL Core	ACK_Re-Transmission_Min		variable
	MPDU_SIZE	1024 o	
	PDU_Delay	0.0 s	
	ACK_PDU_Jitter	1.0 s	
	Min_Scan_Time	60 s	
Source Subnet	Sender Transmission Rate	64 kbps	*10 Mbps*

1. Values in italic overwrite default values shown in Table 7.

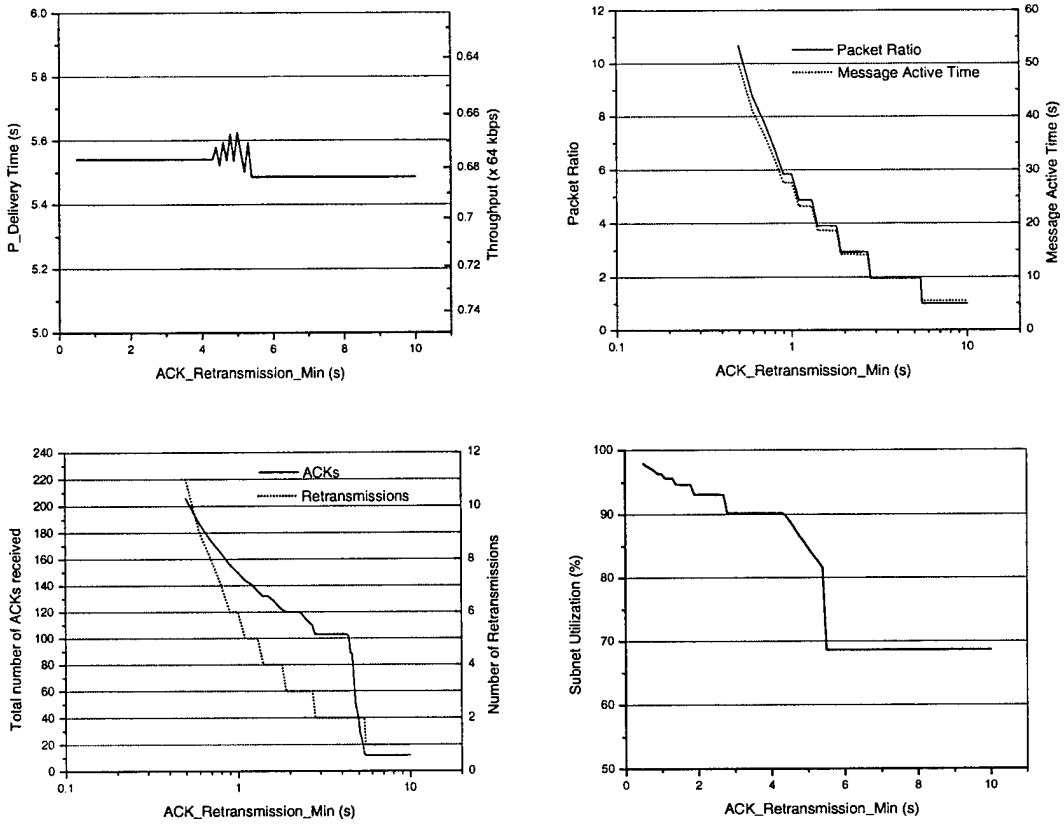


FIGURE 12. Use of the constant ACK Re-transmission scheme to prevent heavy number of retransmissions

The first obvious observation about Figure 12 is that despite the large amount of redundant packets (high packet ratio) being sent towards the destinations, the time to deliver the message is nearly constant over the retransmission range selected, except for a 1 second transition interval where the effect of the random ACK_PDU_Jitter delay can be seen. Interesting as it may be, the response is only partly due to P_MUL. The supplementary packets end up being queued at the destination network. Because of the MAC protocol characteristics, all receivers are given a fair chance of acknowledging the sender (if they need to) between any new broadcast of DATA_PDUs on the subnet.

The transition interval ends when the subnet utilization changes abruptly and stabilises to the 68.6% level as soon as the sender waits a time comparable to the recipients' message reception time before retransmitting the message.

As in scenario A030, the source is sending at a rate that is far too high for the destination it must reach. Interestingly, the ACK_Re-transmission timer appears to be just as "good" if not better than the (static) PDU_Delay parameter to cope with such a non well-behaved source. In both cases, the Message Active Time can be remarkably high when the

source is allow to pump many retransmissions in little time. This is not at all desirable because packets pollute the network long after the message has been fully received and acknowledged (Message Active Time much greater than P_Delivery Time). By setting the ACK Re-transmission timer to a value equal to or greater than the time required to receive the message at the destinations, the overhead is minimised without sacrificing prompt message delivery. This setting corresponds to the second of the two possibilities mentioned at the beginning of this section when discussing what probe to use for estimating the maximum round trip delay.

5.1.3 Progressive Re-Transmissions Scheme

In this section, the set up is similar to scenario A030, except that the progressive retransmission scheme proposed in the Internet-Draft is used to reinitialise the ACK Re-transmission timer. The setting of the PDU_Delay parameter was sampled over a range of 0 to 0.2 seconds, in steps of 0.002 seconds, necessitating 101 simulation runs. The parameter settings are summarised in Table 12 and the results presented in Figure 13.

Table 12. Main parameter settings for scenario A040[1]

	Parameter	Default Value	Scenario A035
P_MUL Core	ACK_Re-Transmission_Min		0.5 s
	ACK_Re-Transmission_Inc		0.5 s
	ACK_Re-Transmission_Max		60.0 s
	MPDU_SIZE	1024 o	
	PDU_Delay		variable
	ACK_PDU_Jitter	1.0 s	
	Min_Scan_Time	60 s	
Source Subnet	Sender Transmission Rate	64 kbps	*10 Mbps*

1. Values in italic overwrite default values shown in Table 7.

Both the constant and the progressive retransmission schemes produce identical P_Delivery Time and Throughput performance. The main difference between the two is in the amount of traffic created on the network. A progressive retransmission scheme reduces the number of retransmissions and the number of acknowledgements without degrading the throughput and response time performance. For instance, for a PDU_Delay setting of zero seconds, the number of redundant retransmissions drops from 11 to 5 whereas the number of acknowledgements goes from 206 to 138. The scheme reduces the amount of pollution on the network (fewer packets wandering around well after the message has been received and acknowledged) as indicated by the much reduced Message Active Time.

The progressive re-transmission scheme provides an effective way to cope with very high data rate senders. The re-transmission timer, whether it uses a constant or a progres-

sive scheme, has the potential of preventing congestion. In a sense, it is a primitive congestion avoidance mechanism which can be quite effective when dealing with fast senders.

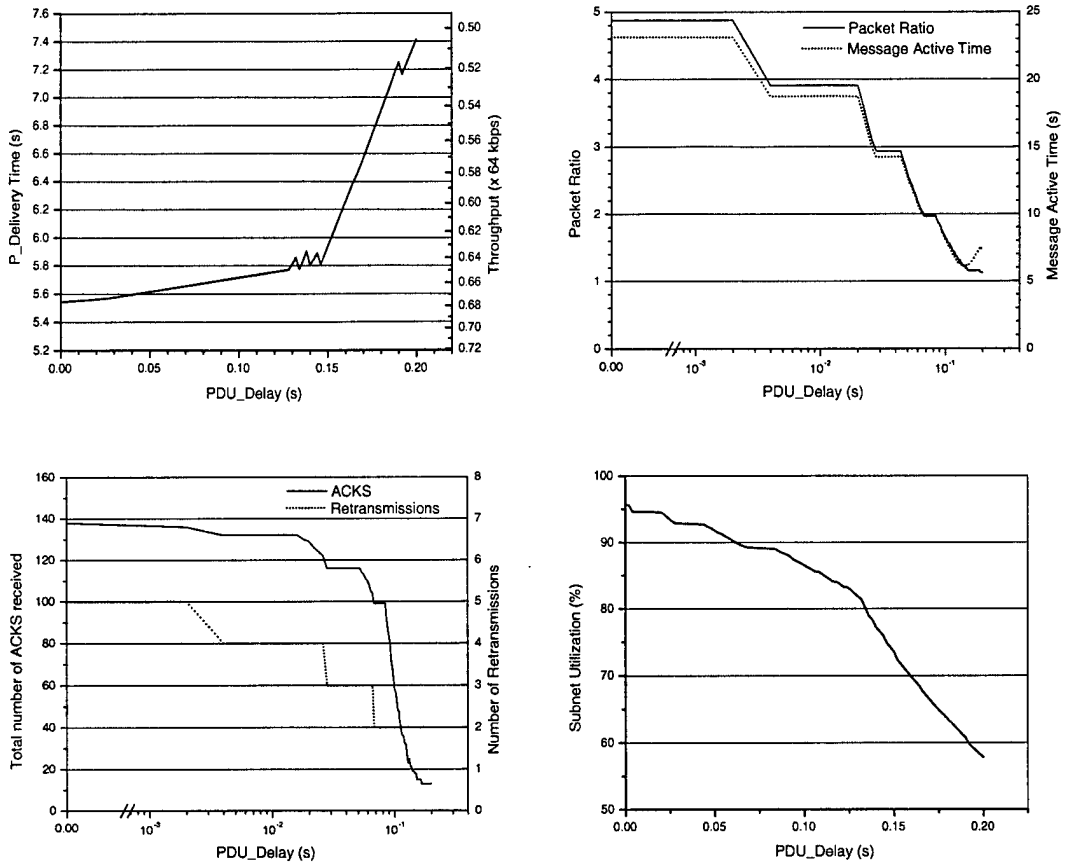


FIGURE 13. Use of the progressive ACK Re-transmission scheme to prevent heavy number of retransmissions

5.1.4 Message Size

Previous simulation scenarios were for a message size of 30 000 octets. A 30 000 byte e-mail message is most often not a small e-mail message whereas a 30 000 byte file message is seldom a large file. It is interesting to ask how well the parameters used in the previous sections fit messages of different sizes. In this section, the message size is varied by a factor of 10 from either side of the value used in the previous sections.

From the results obtained in Section 5.1.2, it is clear that there is a need to investigate further how to choose the initialisation value of the ACK-Re-transmission timer. In this section, two scenarios are set up. One of the scenarios uses an ACK-Re-transmission setting of D_1=0.5 seconds, corresponding to the maximum round trip delay of a maximum size PDU, as calculated by (3) in Section 5.1.2. The other scenario sets the ACK-Re-transmission timer to 6.0 seconds, based on the following maximum round trip delay estimate:

$$D_2 \approx \text{Delivery Delay} + \text{Acknowledgement Delay} + 2 \times \text{Path Delay} + \text{Safeguard}$$

where

$$\text{Delivery Delay} = \frac{8 \times (\text{Message Length} + \text{P\_MUL:UDP:IP:DLL Overhead})}{\text{Nominal Subnet Capacity}} \qquad (5)$$

The Acknowledgement Delay is in the worst case equivalent to the Transmission Delay specified by (4) and can be neglected if the message requires the transmission of more than a dozen or so DATA PDUs. The Path Delay is here again the sum of the propagation delays between the source and the destination that is the most distant in time.

So, in the scenario with D_2= 6.0 seconds, D_2 corresponds to the time required to deliver, at the nominal data rate of 64 kbps, a 30 000 octet message and receive acknowledgement from all destinations plus a generous (about 30%) safeguard.

Both scenarios, A050 and A055, will be presented simultaneously. Table 13 summarises the most relevant parameters. The transmission rate at the sender is normalised to the nominal subnetwork bandwidth (64 kbps) and was sampled over a range of 10 to 100 kbps, in steps of 2 kbps, necessitating 46 simulation runs to obtain every performance curve. The procedure was repeated six times, once for every Message Length setting (3) for both scenarios.

The simulation results are shown in Figures 14 through 16. The plots on the right in Figure 14 were obtained by subtracting the results of the two scenarios. Regardless of the message length, a long (6.0 seconds) retransmission timer causes a slower delivery of the message (1.5 seconds or less) than a short (0.5 second) timer. The maximum difference in throughput is less than 3.1 kbps, i.e. 4.7% of the subnetwork capacity. The best throughput is obtained for large messages and when the destination subnetwork is saturated. However, Figure 15 shows that there is a penalty for overstaturating the network. Oversaturation causes an increase of the Message Active Time (network pollution problem), especially when the message is long and the retransmission timer is short (upper right plot).

Not surprisingly, the worst throughput is obtained for small messages (3 ko). The bottom plots in Figure 15 show that the subnetwork is under-utilized (53% utilization or less), even when the transmission rate of the sender exceeds the subnetwork capacity. This is due to the random delay imposed on the sending of every ACK PDU. The subnetwork is available but no traffic is unicast back to the sender because the ACK_PDU_Jitter timer is delaying the transmission of an acknowledgement. Such a situation could cause problems if several hundreds of short messages needed to be sent in a short period of time.

Table 13. Main parameter settings for scenarios A050 and A055

	Parameter	Default Value	Scenario[1]	
			A050	A055
P_MUL Core	ACK_Re-Transmission_Min MPDU_SIZE PDU_Delay	 1024 o 0.0 s	6.0 s	0.5 s
	ACK_PDU_Jitter Min_Scan_Time	1.0 s 60 s		
Source Subnet	Message Length Sender Transmission Rate	30 ko 64 kbps	*3, 30, or 300 ko* *variable*	*3, 30, or 300 ko* *variable*

1. Values in italic overwrite default values shown in Table 7.

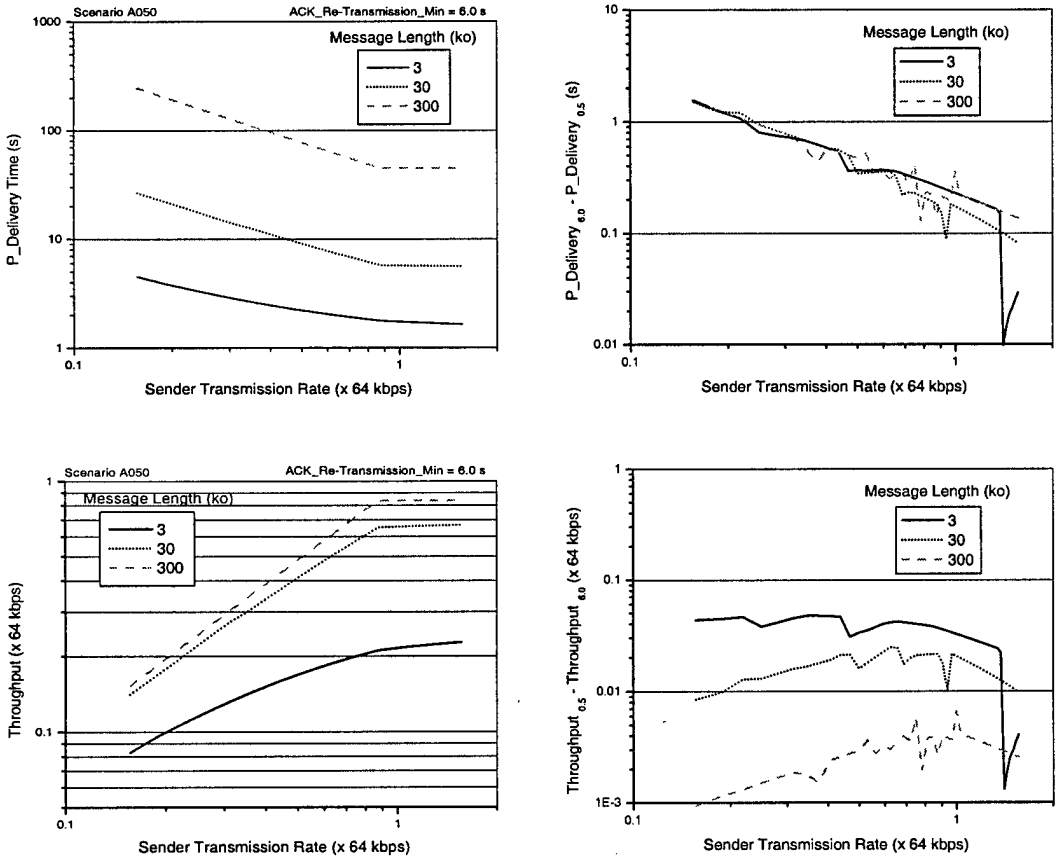


FIGURE 14. Effect of message length on performance

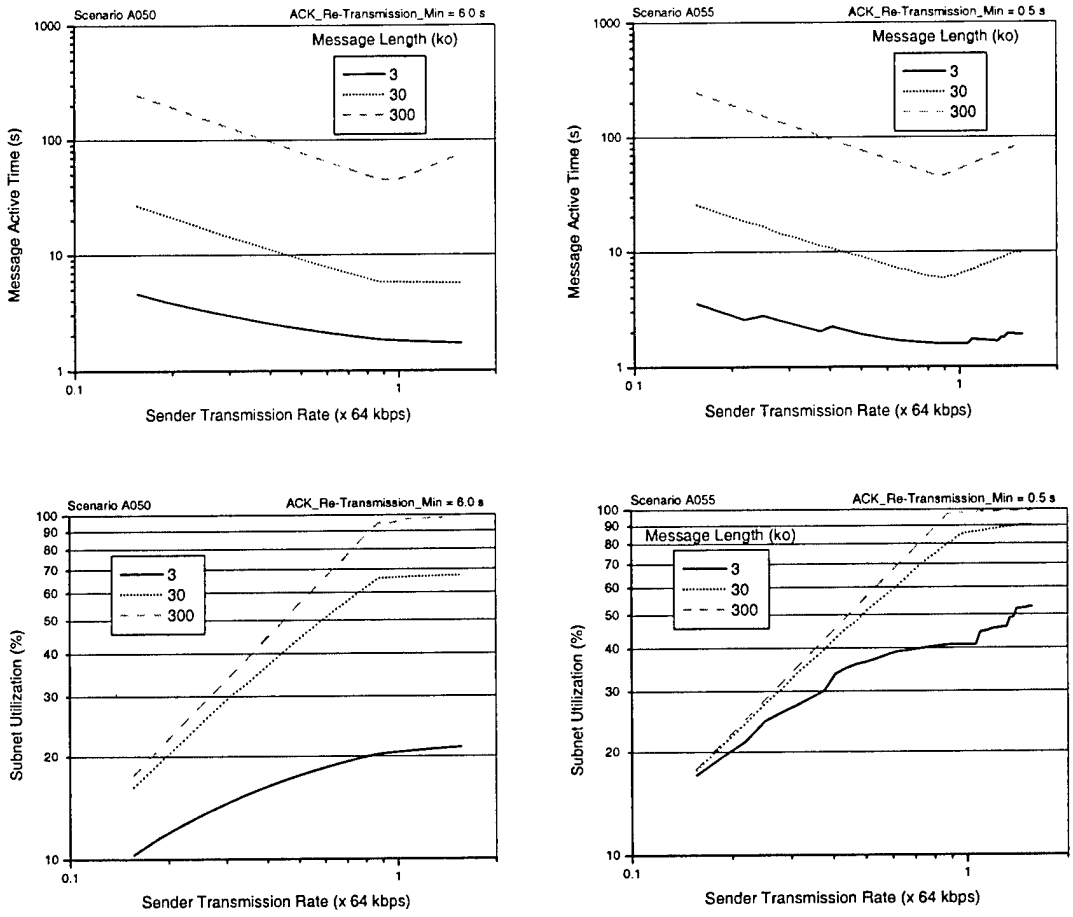


FIGURE 15. Effect of message length on performance (continued)

The remaining results are shown in Figure 16. The short retransmission timer causes more retransmissions than the long timer, even when the transmission rate at the sender is much lower than the capacity available at destination. Large messages will cause many acknowledgements to be received if the ACK Re-Transmission timer is set to accommodate a message of average size.

Overall, performance achieved with the long (6.0 s) retransmission timer is better than that obtained with the short (0.5 s) timer, especially when the sender data rate nearly matches the nominal capacity of the destination network.

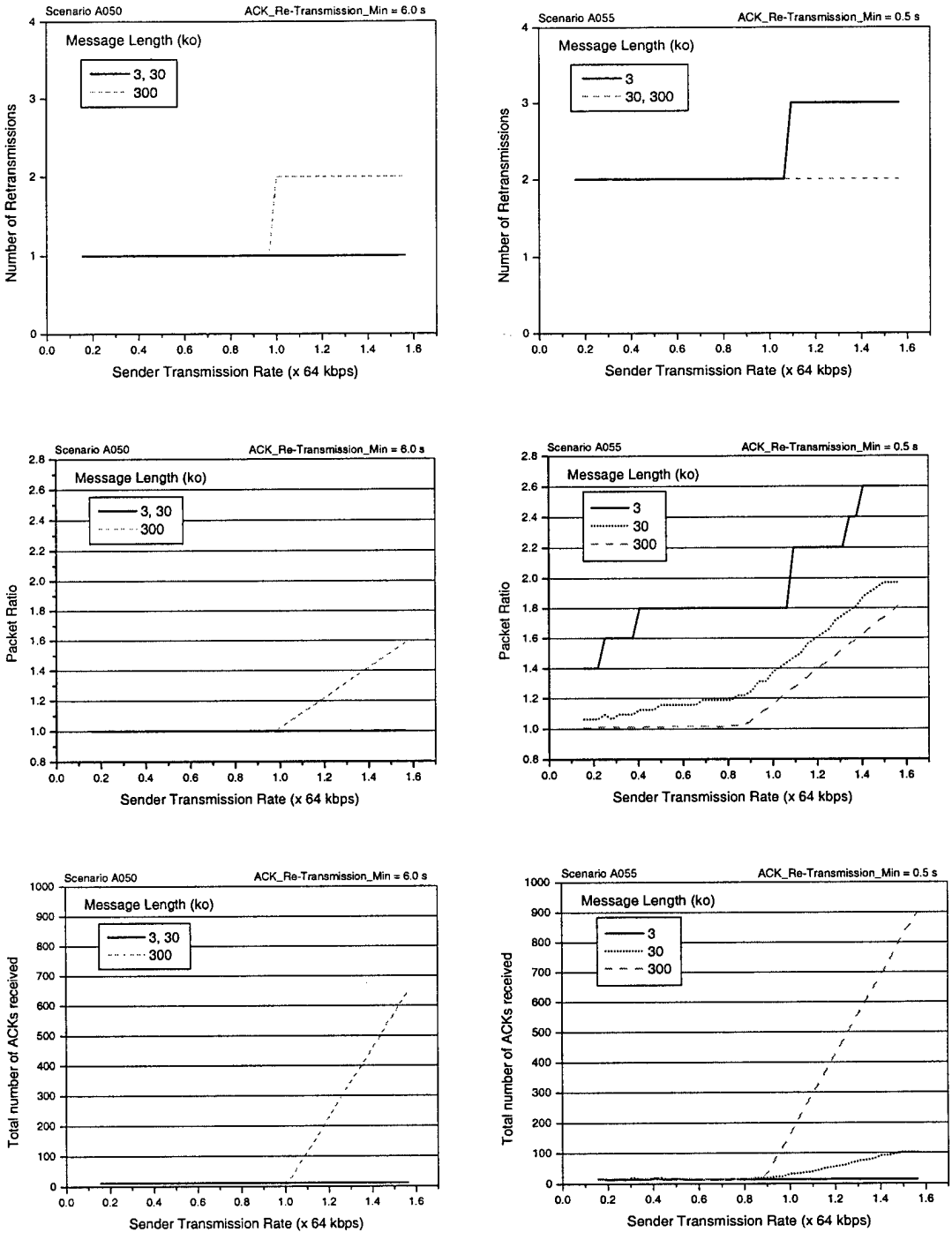


FIGURE 16. Effect of message length on performance (continued)

5.1.5 PDU Size

Previous simulation scenarios were set up to transmit DATA_PDUs not exceeding 1024 octets in size. In this section, the message size is constant at 30 000 octets but the maximum DATA_PDU size is varied between every simulation runs. The MPDU_SIZE parameter is configured for either 256, 512, 1024 or 2048 octets. As in Section 5.1.4, two scenarios are set up simultaneously to compare the impact of a long (6.0 seconds) and a short (0.5 second) retransmission interval.

Parameters for both scenarios, A060 and A065, are summarised in Table 14. The normalised transmission rate at the sender is once again taken as being the independent variable, every characterisation curve produced requiring 46 simulation runs. The procedure is repeated eight times, once for every MPDU_SIZE setting (4) of the two scenarios.

Table 14. Main parameter settings for scenarios A060 and A065

	Parameter	Default Value	Scenario[1]	
			A060	A065
P_MUL Core	ACK_Re-Transmission_Min		6.0 s	0.5 s
	MPDU_SIZE		2^8, 2^9, 2^{10} or 2^{11} o	2^8, 2^9, 2^{10} or 2^{11} o
	PDU_Delay	0.0 s		
	ACK_PDU_Jitter	1.0 s		
	Min_Scan_Time	60 s		
Source Subnet	Message Length	30 ko		
	Sender Transmission Rate	64 kbps	*variable*	*variable*

1. Values in italic overwrite default values shown in Table 7.

Selected simulation results are shown in Figures 17 and 18. Varying the maximum size of the P_MUL packets does not significantly affect the response time. For both the P_Delivery and Throughput metrics results obtained for the long and short retransmission timers are so similar that results for only one of the two settings are presented in Figure 17. The difference in delivery time was less than a third of a second whereas the difference in throughput was less than 2.2 kbps or 3.4% of the subnetwork capacity.

The small packets definitely create a higher demand on the destination subnetwork utilization than the large packets do. The utilization levels are the highest for the short retransmission timer and when the sender saturates the network. As shown in previous sections, a short retransmission interval does not provide any major benefit, quite the opposite. The Message Active Time metric shows that there is network pollution if the sender data rate is faster than the nominal capacity of the destination network. In the case of the long retransmission interval, the sender has time after sending the message once to receive the necessary (12) acknowledgements from all (12) receivers before the timer expires. Figure 18 shows that up to three retransmissions are occurring when the retrans-

mission timer is short and the maximum packet size is 256 octets. In this particular case, this translates into a doubling of the number of packets transmitted by the sender (Packet Ratio of 2) and an increase in the number of acknowledgements by a factor of 12 (from 12 to 145).

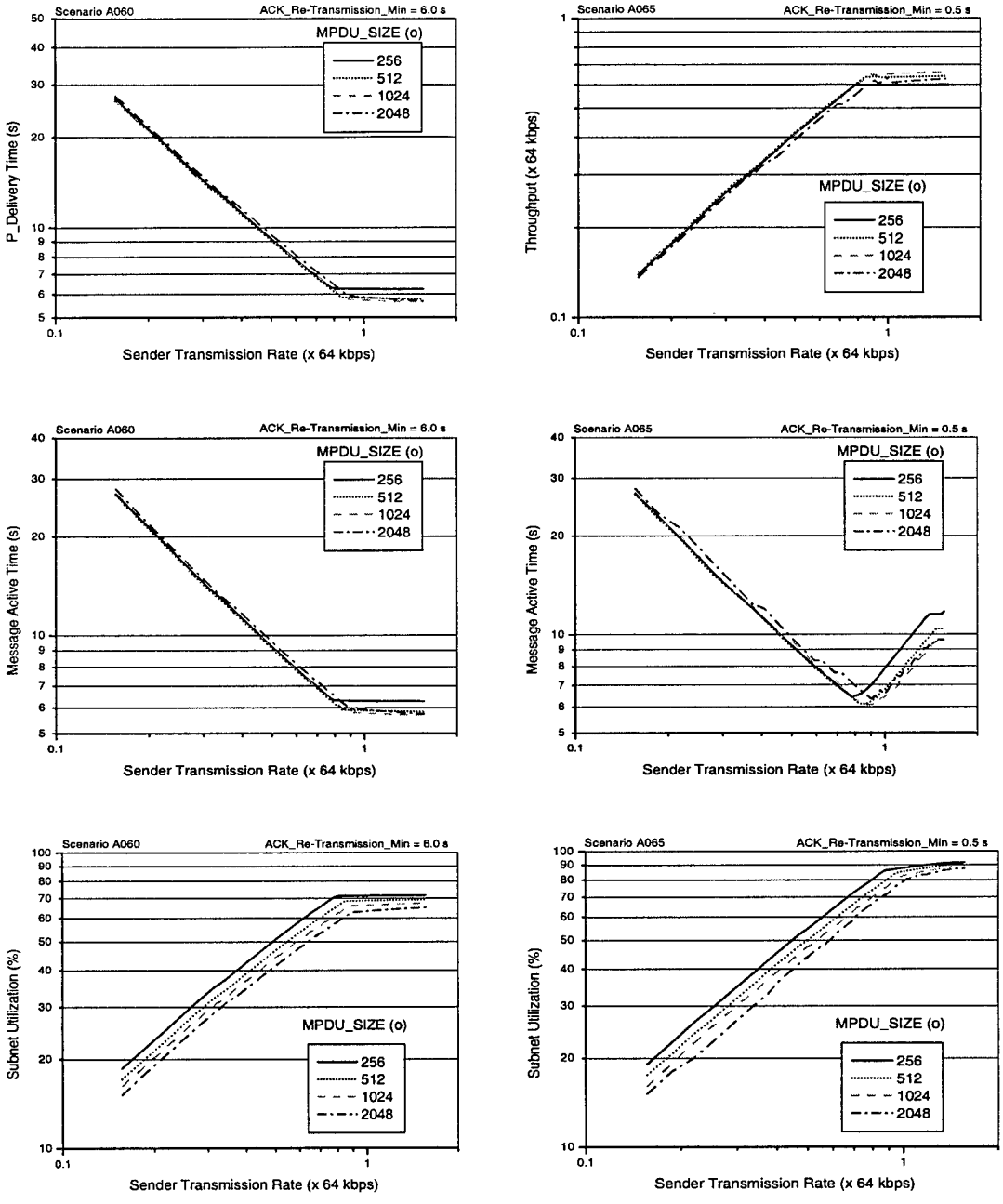
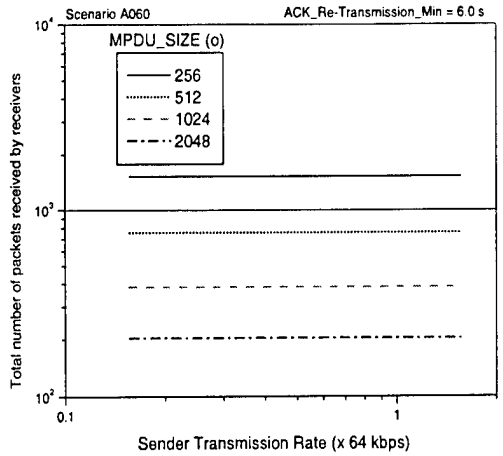
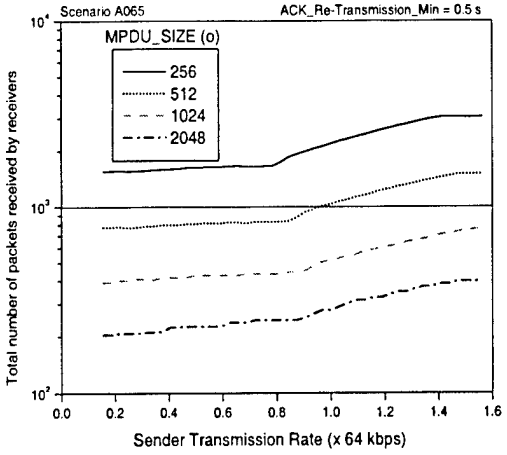
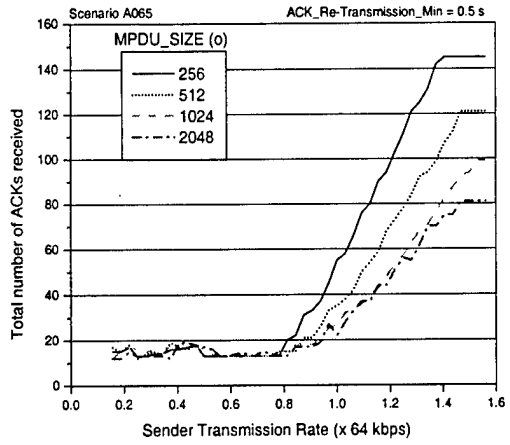
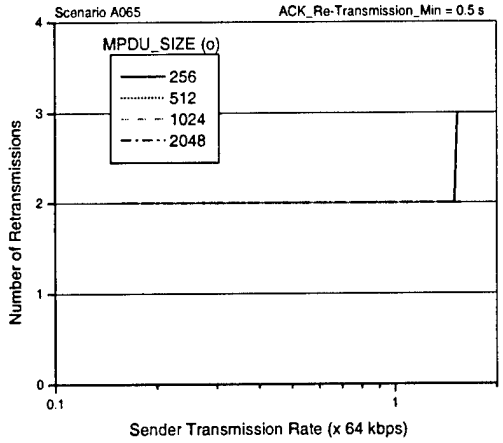


FIGURE 17. Effect of maximum PDU size on performance









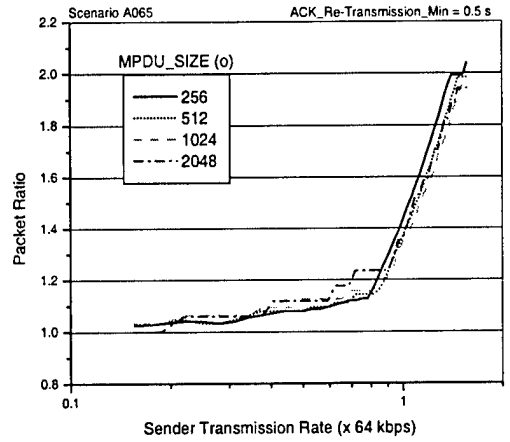


FIGURE 18. Effect of maximum PDU size on performance (continued)

5.1.6 Dynamic Group Installation

P_MUL provides the capability of dynamically creating multicast groups on a per-message basis. However, dynamic address negotiation can be costly. Address negotiation introduces overhead independent of the length of the message.

The overhead introduced by address negotiation is due to predefined delays at the T-node requesting the address. These delays are:

1. A delay of WAIT_FOR_REJECT_TIME seconds after the transmission of the REQUEST_PDU; and,
2. A delay of ANNOUNCE_DELAY seconds following the transmission of each set of ANNOUNCE_PDUs.

The total delay introduced can therefore be expressed as:

$$\text{neg\_delay} = \text{WAIT\_FOR\_REJECT\_TIME} + (\text{ANNOUNCE\_ct} + 1) \times \text{ANNOUNCE\_DELAY} \quad (6)$$

The ACP does not recommend nor provide much indication on how to adjust the parameters in (6) for best results. The value of WAIT_FOR_REJECT_TIME must be large enough so that each T-node has time to receive the REQUEST_PDU and send a REJECT_PDU if necessary. Therefore, WAIT_FOR_REJECT_TIME should be set to the round trip delay for a *group negotiation PDU* plus a safeguard.

Similarly, it is important that all receivers receive the ANNOUNCE_PDUs before message transmission begins. ANNOUNCE_PDUs are retransmitted ANNOUNCE_ct times, with each retransmission separated by ANNOUNCE_DELAY seconds. Since no response is required to the ANNOUNCE_PDUs a value equal to the round trip delay should be sufficient for the ANNOUNCE_DELAY parameter. The value of ANNOUNCE_ct should be chosen based on the expected link quality. In high error rate environments the ANNOUNCE_PDUs should be retransmitted several times to increase the likelihood that they are received by all intended recipients.

For short messages, the time required to negotiate a multicast address can be much greater than the length of time required to transfer the message. To maximize the throughput of a network, multicast address negotiation should in general be reserved for messages where:

$$\text{neg\_delay} \ll \text{data transfer time} \quad (7)$$

Figure 19 shows an example of the number of octets that must be transferred for the address negotiation time to make up a significant portion (33%) of the total message transfer time for various round trip delay values. Calculations for this graph were made based on a message destined for six receivers, with a nominal subnet data rate of 64 kbps, a maximum PDU size of 64 octets, and 96 bits plus 10% overhead per packet. The value of

ANNOUNCE_ct is 0 for the bottom curve, 2 for the middle curve and 4 for the top curve. The figure demonstrates that even for situations where the round trip delay time is small, the message must be several thousand octets in length before the proportion of time taken for address negotiation can be neglected.

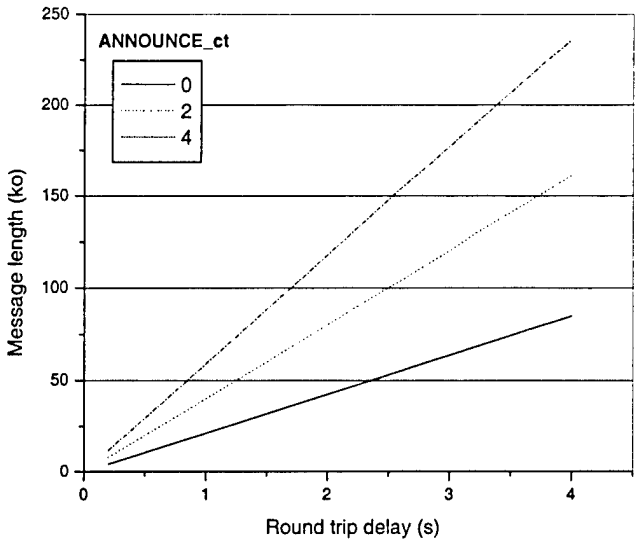


FIGURE 19. Message length for which negotiation time is 33% of the total message transfer time

Because negotiation is done on a per message basis, metrics like P_Delivery Time and Throughput are measured from the Message Start time as it is being done when the group addressing is static. It would make sense to disregard the time to negotiate a multicast address if this were done once and the installation would remain effective for any new message sent by the sender. But that is not the case. Every message requires a new negotiation and this does not come free; in the CRC model, the price to pay is reflected in the achievable throughput and time to complete a message transfer. If the sender needs to send thousands, hundreds, or perhaps even tens of short messages then dynamic group installation is a convenience that can be costly.

5.1.7 ACK Implosion

In the P_MUL protocol there are five reasons for sending an ACK_PDU:

1. More than some number M_Missing_DATA_PDUS are missing;
2. More than Min_Scan_Time has elapsed without a PDU having been received for this message (strictly speaking, this parameter is not defined in the ACP so the Scan Timer may not be part of every implementation);
3. The last DATA_PDU has been received;

4. A duplicate DATA_PDU has been received while the receiver is waiting for a complete message transfer acknowledgement (ADDRESS_PDU) from the sender;
5. The ACK_PDU timer has elapsed and none of the expected DATA_PDUs has been received (only applies to receivers that have just come out of EMCON).

When many receivers independently and nearly simultaneously choose to send ACK_PDUs to one sender, the sender may be flooded with more ACK_PDUs than it is capable of handling. Such situation is known as the *ACK implosion* problem. Two examples among others are when a group of receivers receives the final PDU (case 3 above), or when a group of EMCON receivers return to non-EMCON mode (case 5). ACK implosion could also occur when the link quality is poor and many PDUs are lost, causing receivers to send several ACK_PDUs at once.

The P_MUL protocol uses the ACK_PDU_Jitter timer to stem the output of ACK_PDUs at the receivers. The value used to initialise the ACK_PDU_Jitter timer is implementation dependant and either a constant value or a random value may be used. The ACP 142 suggests that a random timer be used to avoid ACK implosion. A random value for the ACK_PDU_Jitter timer will help to ensure that a large group of different receivers do not simultaneously send ACK_PDUs. However, a purely random delay would permit in rare occasions a particular receiver to send ACK_PDUs with almost no delay between them. Also, a random delay time does not guarantee a delay of at least the maximum round trip time between the transmission of ACK_PDUs, meaning that several ACK_PDUs could be sent before the sender has time to respond.

The likelihood of ACK implosion at the sender and the effectiveness of the ACK_PDU_Jitter timer as a method of avoiding it needs to be further investigated.

5.2 Unreliable Network Service

One of P_MUL's objectives is the reliable data transfer of messages. Since P_MUL uses a connectionless transport protocol to transfer messages over multicast subnetworks, the protocol must be robust to combat adverse channel conditions and quick to recover from bit corruption (data errors) introduced in transmissions. Section 5.2 analyses how P_MUL reacts when some of the data and control packets are lost during the course of a message transfer.

5.2.1 Probability of Error

If we disregard for a moment any techniques used to improve the transfer of digital information across communications links, the probability that errors occur when bits of information are sent across an information channel solely depends on the channel's bit error rate (*ber*). A very simple way of determining the number of bit errors within a given packet is to examine each bit within the packet and decide the correctness of each bit based upon the probability of a bit error. In a packet of length N, the probability that any

one bit has an error is equal to the ber of the link. The probability P_0 that no errors occur corresponds to N successive events with probability (1-*ber*) and is given by the following expression:

$$P_0 = (1 - ber)^N \tag{8}$$

The probability P_e that one or more errors occur is:

$$P_e = 1 - P_0 = 1 - (1 - ber)^N \tag{9}$$

Given that P_MUL has no built-in error correction mechanism, a packet must be rejected by an R-node receiver if it contains one or more errors. For this reason, expression (9) can directly be applied to the packet structure defined for P_MUL. Figure 20 shows the expected probability P_e of rejecting a packet of size N bits, where $N = 8 \times$ MPDU_SIZE.

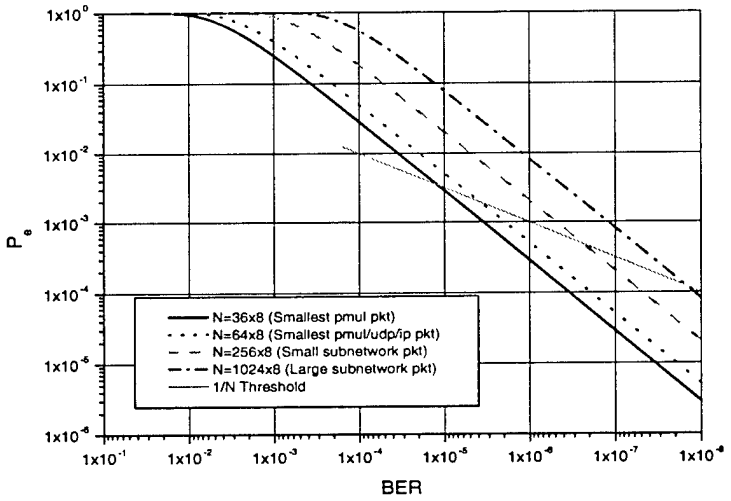


FIGURE 20. Probability of P_MUL packet rejection in noisy channel

The first curve, for N=36x8 bits, corresponds to the best link error resilience achievable by P_MUL. It is obtained with a maximum packet size of 36 bytes. Why 36? Indeed, better error tolerance would be achieved if the packet size could be made smaller. However, the value 36 is an intrinsic lower limit imposed in the ACP 142 to the ADDRESS_PDU. Of all the packet formats defined in the ACP 142, the ADDRESS_PDU is the one that has the largest header. When restricted to a size of 36 bytes, the ADDRESS_PDU can only flow control one destination at a time, which is also the minimum requirement to make P_MUL work. By setting MPDU_SIZE to 36 bytes, every DATA_PDU will therefore carry a maximum payload of 20 bytes of information, since 16 bytes are needed for the header.

The performance with N=36×8 bits are rather academic since in practice P_MUL requires transport and network layers to operate. The second curve in Figure 20, for N=64×8 bits, corresponds to the best link error resilience achievable by P_MUL when the UDP and IP overhead are included in the total packet size N. That is, 36 bytes for the P_MUL PDU, 8 bytes for the UDP header and 20 bytes for the IP header, for a total of 64 bytes altogether. It is justifiable to make use of (9) with the UDP/IP overhead tacked-on to the P_MUL PDU because neither UDP nor IP performs any error correction on the packet and both will reject it if it contains one or more errors.

The last two curves in Figure 20 are examples of larger N values where some of the bits in the transmitted packet are perhaps allocated to subnetwork overhead and some more to increase the P_MUL payload beyond the 20 bytes minimum already discussed. One thing to remember when interpreting these curves is that nowhere in the layer stack is error correction performed and the entire packet is rejected as soon as an error is detected through the usual checksum verification.

5.2.1.1 Note on Maximum Message Size

The message size is limited by the `Total_Number_of_PDUs` field of the `ADDRESS_PDU` and the size of the `DATA_PDU`. The size of `Total_Number_of_PDUs` field is 2 octets which corresponds to a maximum of 2^{16} PDUs. For a maximum `DATA_PDU` of size `MPDU_SIZE` = 36, there is a maximum of 36-16=20 data bytes per `DATA_PDU`, since every `DATA_PDU` has a 16 byte header. The maximum message size for a minimum size PDU is 2^{16}×20 or about 1.3 Mo.

5.2.2 Feedthrough and Null Subnets

The topology shown in Figure 21 will be used to analyse P_MUL in a noisy environment. The topology is configured to bypass any overhead introduced by UDP, IP, and the MAC protocol. By setting the proper parameters, this interesting capability allows us to investigate P_MUL as if a peer to peer packet transfer between nodes was directly made at the P_MUL layer level. Any errors introduced over the communication links end up being directly injected into P_MUL packets. This configuration can provide very basic performance information that cannot be reproduced in a real environment but that is very valuable from a theoretical point of view.

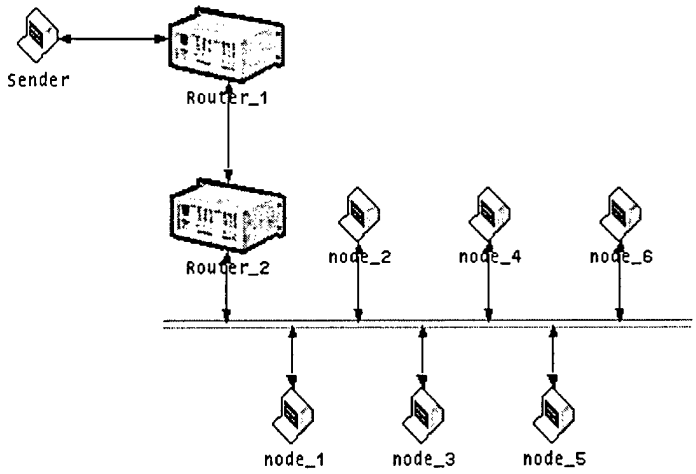


FIGURE 21. Simulation topology B

The topology is composed of one T-node (sender), one high-speed (10 Mbps) zero delay (0 ms) point-to-point transit link between the source and destination routers and a 64 kbps destination broadcast network where resides a multicast group of 6 recipients (node_1 to node_6). It is assumed that all nodes are operating in the non-EMCON conditions. The communication channels are noisy, some bit errors being introduced randomly (uniform distribution) to simulate unreliable links. The default UDP, IP and DLL protocol overhead have been set to zero. Other environment parameters most relevant to the scenarios of this section (Section 5.2) are summarised in Table 15.

Table 15. Default parameter settings for simulation topology B

	Parameter	Default Values
Source Subnet	Message Length	30 ko
	Sender Transmission Rate	64 kbps
	Source Data Rate	64 kbps
	Source ptp_overhead_abs	0 bits
	Source ptp_overhead_rel	0%
	Source BER	0 error per bit
Transit Subnet	Transit Data Rate	10 Mbps
	Transit Delay	0 s
	Transit ptp_overhead_abs	0 bits
	Transit ptp_overhead_rel	0%
	Transit BER	0 error per bit
Destination Subnet	Destination Data Rate	64 kbps
	Destination bus_overhead_abs	0 bits
	Destination bus_overhead_rel	0%

5.2.3 Performance with Maximum PDU Size of 1024 Octets

The probability of P_MUL packet rejection computed in Section 5.2.1 indicates that the default packet size of 1024 octets used during the introductory scenarios in Section 5.1 is not likely to provide good message transfer performance. Nevertheless, it is interesting to verify whether this is really the case before searching for the optimum packet size. This analysis begins with two scenarios, one with an inter-retransmission interval of 6.0 seconds, the other with a 0.5-second interval. In each case, the message must be transferred to the unreliable multicast subnetwork shown in the network topology of Figure 21 using the parameter settings listed in Table 16. The bit-error-rate is typically set to five different values per decade, i.e., on values equal to 1, 1.8, 3, 5 and 7. A time limit of four hours has been set for the delivery of the message. Given that under error-free conditions, the short 30 000 octet message should be received in less than 5 seconds, a message expiry time of four hours is ridiculously high. If the transfer fails after trying for 4 hours, chances are the protocol would not be of much use in practice.

Table 16. Main parameter settings for scenarios B010 and B015

	Parameter	Default Value	Scenario[1]	
			B010	B015
P_MUL Core	ACK_Re-Transmission_Min		6.0 s	0.5 s
	MPDU_SIZE	1024		
	Delete_DATA_PDUS_Time	60 s		
	M_Missing_DATA_PDUS	8 pdus		
	ACK_PDU_Jitter	1.0 s		
	Min_Scan_Time	60 s		
Source Subnet	Message Expiry Time	4 hr		
Dest. Subnet	Destination BER		*variable*	*variable*

1. Values in italic overwrite default values shown in Table 15.

The protocol's performance under noisy conditions is shown in Figures 22 and 23. The message transfer starts to fail around 10^{-3} errors per bit where at best 4 out of the 6 receivers managed to acknowledge the correct receipt of the message.

Most curves for BER greater than 10^{-3} are truncated at the top because the message transfer was aborted when the message expiry time was reached four hours after the start of the transmission.

There is a note in the Internet-Draft (Section 5.1.3.1) saying that for lossy networks duplicate transmission of the ACK_PDUs is recommended since the [missed] reception of

the ACK_PDUs bears a strong impact on the overall network performance. The bottom plots in Figure 22 indicate that this alone is perhaps not sufficient as there comes for each scenario a point where the number of acknowledgements received in error exceeds the number of error-free acknowledgements. In the present case, this occurs around 2.2×10^{-3} errors per bit, with about 8400 acknowledgments recorded in scenario B010 and 7000 in scenario B015.

In scenario B010, the Subnet Utilization plot shows that there is lots of dead time were the subnetwork is not being used. This is no longer true in scenario B015. In fact, for a BER of 10^{-3} or more, the number of retransmissions is on average twice that of scenario B010 (3600 versus 1800 retransmissions).

So, neither increasing the number of retransmissions nor the number of acknowledgement is likely to provide much performance gain. It would appear that the solution lies in reducing the packet size as foreseen in Section 5.2.1.

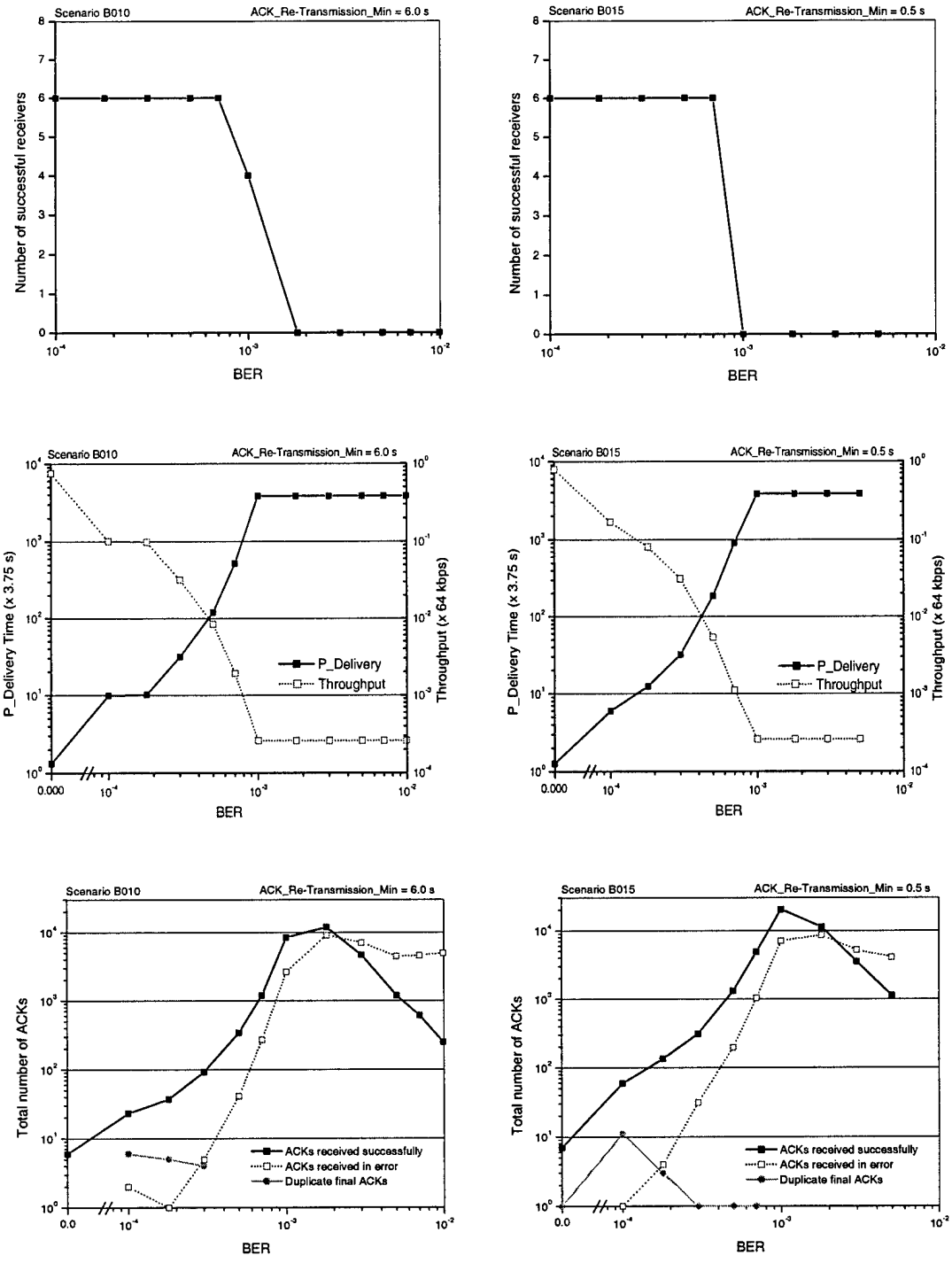


FIGURE 22. Performance with MPDU_SIZE set at 1024 octets

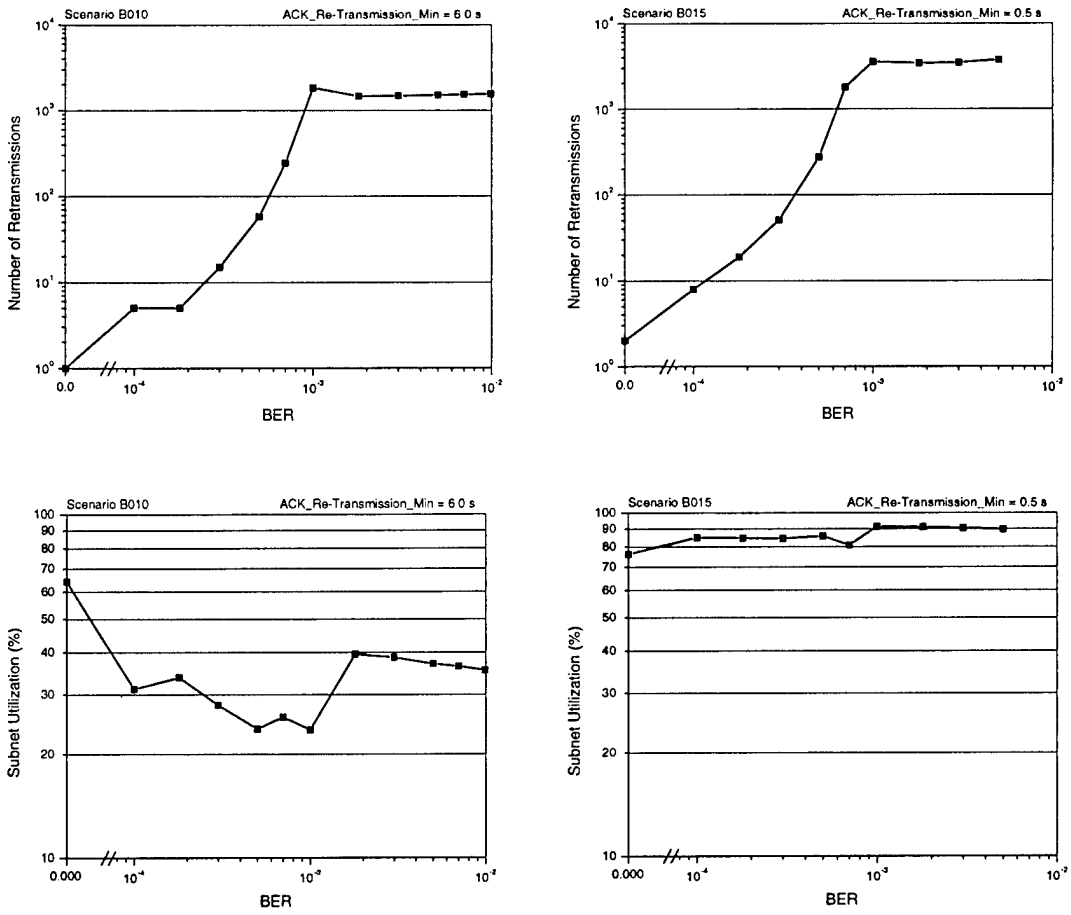


FIGURE 23. Performance with MPDU_SIZE set at 1024 octets (continued)

5.2.4 The Maximum PDU Size Parameter

Figure 20 indicates that the probability of rejection of P_MUL packets due to transmission errors is theoretically the lowest when packets are made as small as possible. As discussed in Section 5.2.1, the smallest value imposed by the ACP 142 on the predefined MPDU_SIZE parameter is 36 octets. This corresponds to a P_MUL data packet composed of a 16 byte header and a 20 byte payload. Such a small packet has 44.4% overhead, which is very substantial considering that additional overhead for the transport, network and data link layers must be appended when using the protocol in a real functional environment.

In this section, each scenario scans the MPDU_SIZE parameter from 36 to 1024 in steps of 10 octets. The bit error rate at the destination is kept constant during each scenario and set to a value of 10^{-3} error per bit. A total of 100 simulation runs are required to produce every characterisation curve. Two scenarios are set up simultaneously to compare

once again the impact of a long (6.0 seconds) and a short (0.5 second) retransmission interval. Parameters for both scenarios, B020 and B021, are summarised in Table 17.

Table 17. Main parameter settings for scenarios B020 and B021

	Parameter	Default Value	Scenario[1]	
			B020	B021
P_MUL Core	ACK_Re-Transmission_Min		6.0 s	0.5 s
	MPDU_SIZE		*variable*	*variable*
	Delete_DATA_PDUS_Time	60 s		
	M_Missing_DATA_PDUS	8 pdus		
	ACK_PDU_Jitter	1.0 s		
	Min_Scan_Time	60 s		
Source Subnet	Message Expiry Time	4 hr		
Dest. Subnet	Destination BER		10^{-3}	10^{-3}

1. Values in italic overwrite default values shown in Table 15.

Selected simulation results are shown in Figures 24 and 25. The message transfer starts to fail when the maximum P_MUL packet size exceeds 706 and 866 octets for the long and short retransmission intervals respectively. Unlike scenarios A060 and A065 in Section 5.1.5, where the noise component was absent, the size of the P_MUL packets strongly impacts the response time of the message transfer. In the presence of noise, the time to deliver the message increases, for the most part, linearly with the logarithm of the P_MUL packet size. Also, the response time is much faster for the short retransmission interval than it is for the long interval. In both cases, the curve flattens out at the top when the message expiry time has been reached four hours after the start of the transmission. Both scenarios generate roughly the same total number of acknowledgements, up until the simulation runs begin to abort. Both the number of acknowledgements and retransmissions also increase linearly with the logarithm of the P_MUL packet, so, presumably, the linear relationship between the delivery time and the maximum packet size could be used to estimate what length of time the message expiry time must be set at to successfully deliver the message with a maximum packet size exceeding the cutoff where the message transfer aborts.

The number of retransmissions is more numerous for the short than for the long retransmissions scenario. This is reflected in the utilization level of the destination network, which is for a wide range of packet sizes being used about 4 times more (80% versus 20%) during the short retransmissions scenario than during the long scenario. However, there are many instances during both scenarios where the utilization level of the destination network goes down near 0%. These odd results exactly correlate with the sudden impulses seen in the Message Active Time plots at the bottom of Figure 25. A similar

impulse can also be observed in the P_Delivery plots in Figure 24 for MPDU_SIZE equal 46 octets. The origin of these glitches has been traced and attributed to the loss of some P_MUL control packets during the transfer of the messages. A detailed explanation of these problems is presented in Section 5.2.4.1.

Because of the fairly high variance in the results, it is not clear whether the linear characteristics discussed above are maintained when the maximum packet size is at or near its lowest admissible values. To clarify the matter, the same two scenarios were repeated with a small MPDU_SIZE increment of 2 octets with 10 runs for each setting. This procedure improves the resolution and reduces the statistical variance. The random noise generator was re-initialised with a different seed between each simulation run. The sampling of the MPDU_SIZE was limited to the range 36 to 256 octets.

The results for these additional runs are shown in Figures 26 and 27. A degradation in performance can be observed when the MPDU_SIZE is at its lowest values. The throughput curves in particular clearly show a maximum when the MPDU_SIZE is set at 48 and 56 octets for the long and short retransmission intervals respectively.

A comparison between the two scenarios for selected MPDU_SIZEs of 36, 64, 128, 256 and 512 octets is offered in Table 18. The results are sorted by order of best delivery time performance. The short retransmission scenario occupies the first 3 positions and provides the fastest response time. The first 5 positions are occupied by scenarios where the maximum packet size is 128 octets or less. It is interesting to note that the long retransmission scenario has the second and third best Message Transfer Efficiency and the lowest Packet Ratio of all. Its use of the subnetwork is low (about one third lower than the one of the fastest performer) because of the relatively few retransmissions but it nevertheless completes the message transfer relatively quickly (in about three times longer than the fastest performers but over 28 times faster than the slowest performer).

Table 18. Performance comparison between scenarios B020 and B021

Scenario	MPDU_SIZE (octet)	P_Delivery (x 3.75 s)	Throughput (x 64 kbps)	Utilization (%)	Packet ratio	ACKs	MTE (%)
B121	64	7.7	0.1305	62.9	3.5	203.6	20.8
B121	128	10.7	0.0934	72.8	6.7	234.1	12.8
B121	36	11.8	0.0847	56.0	3.0	311.3	15.1
B120	36	22.4	0.0446	22.7	2.6	340.3	19.7
B120	64	27.5	0.0364	18.9	4.0	254.8	19.2
B121	256	27.7	0.0362	75.1	18.9	501.9	4.8
B120	128	40.5	0.0247	21.0	6.3	300.5	11.8
B120	256	101.2	0.0099	20.8	18.9	565.2	4.7
B121	512	208.5	0.0048	77.4	153.9	1731.8	0.6
B120	512	786.7	0.0013	20.3	147.1	1737.0	0.6

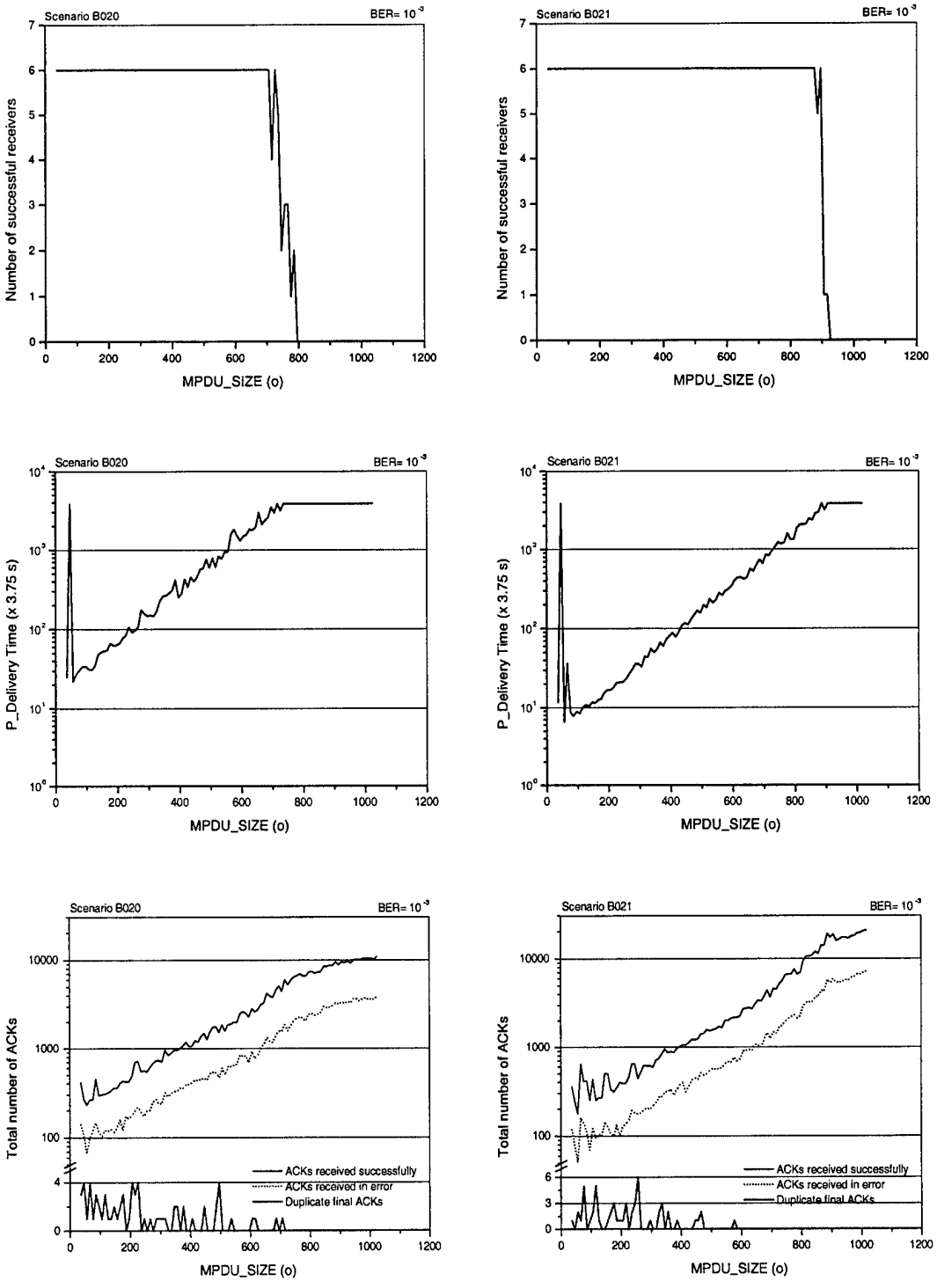


FIGURE 24. Effect of maximum PDU size on performance

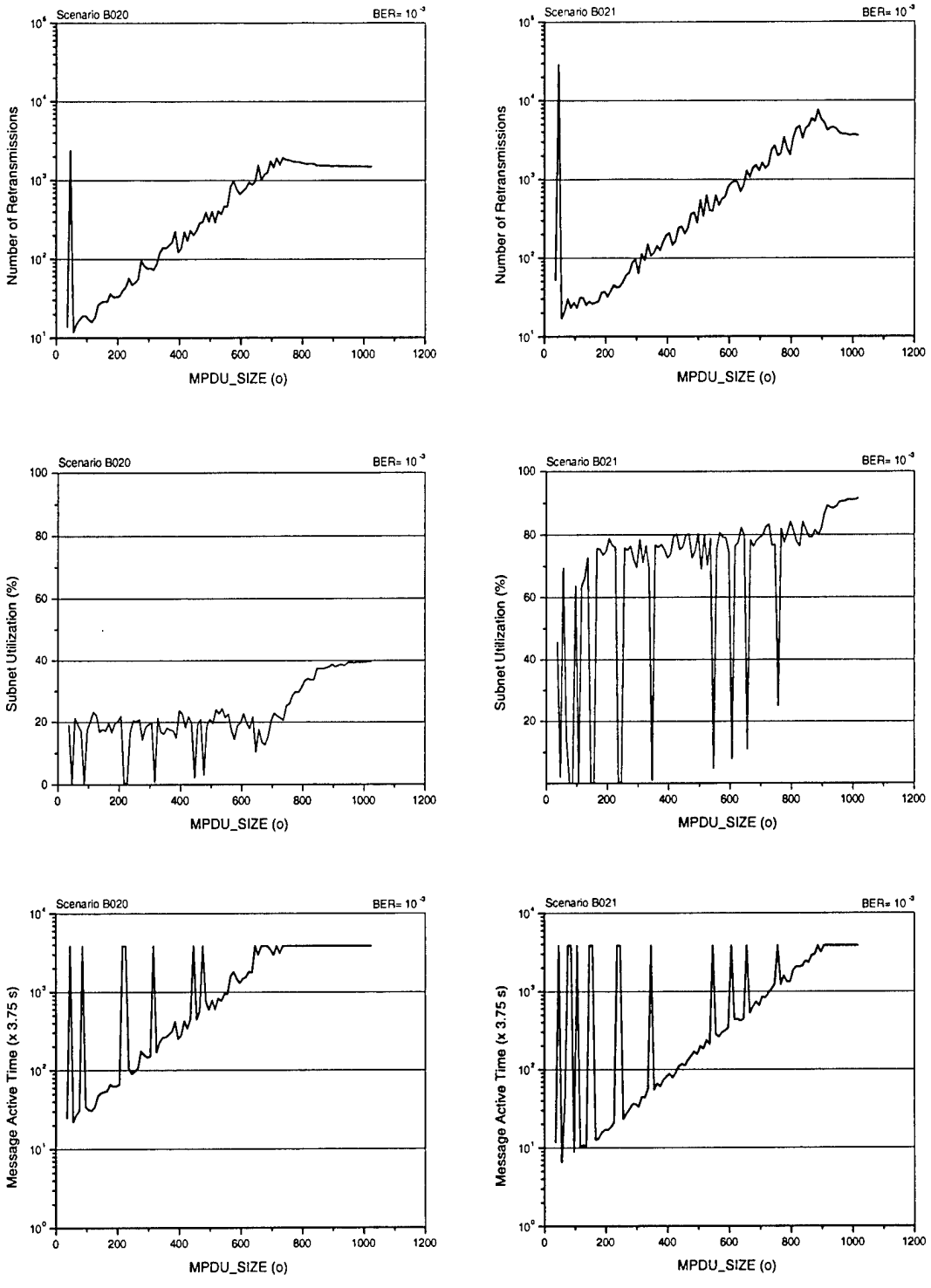


FIGURE 25. Effect of maximum PDU size on performance (continued)

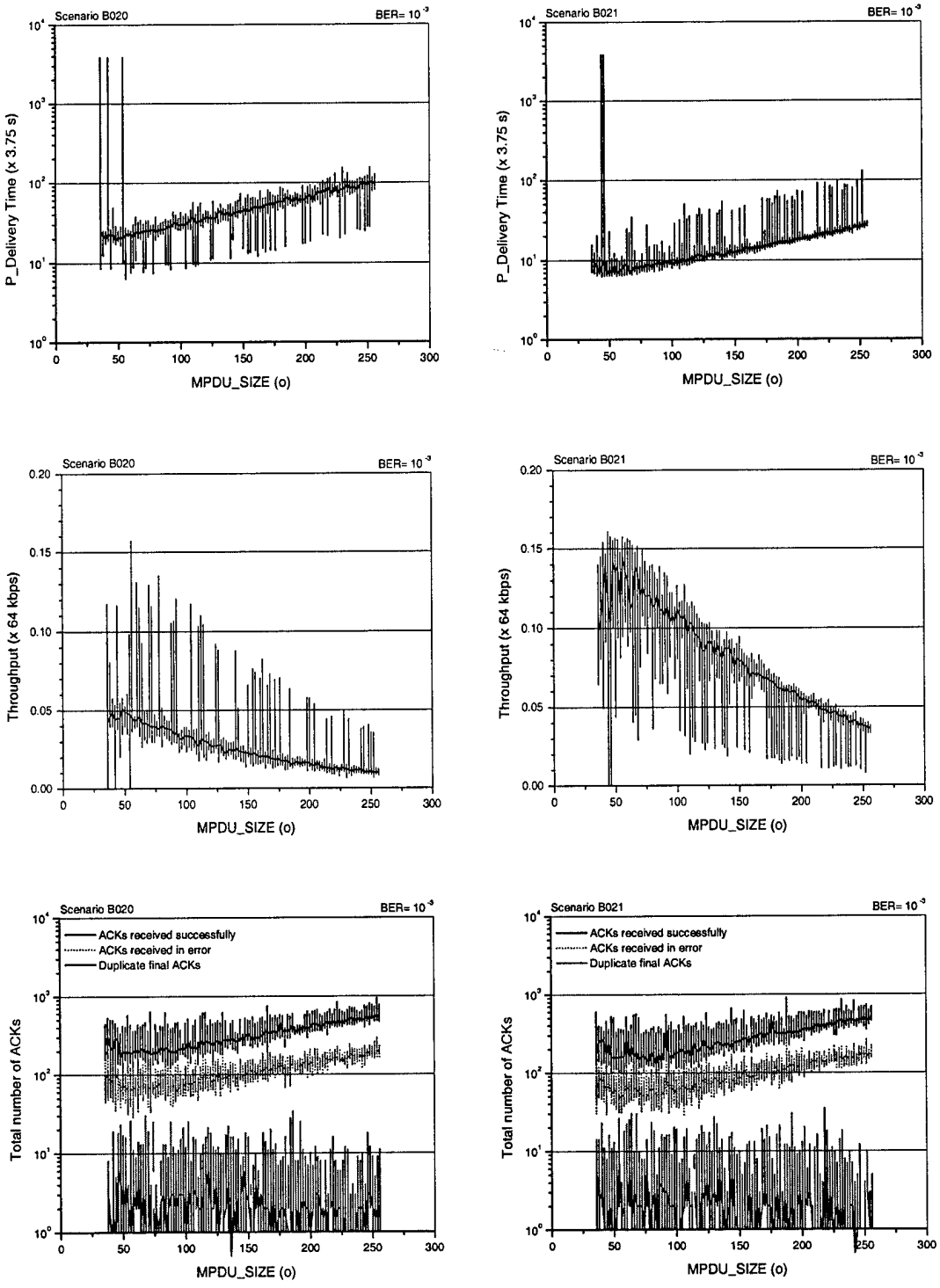


FIGURE 26. Performance for the lower MPDU_SIZE range

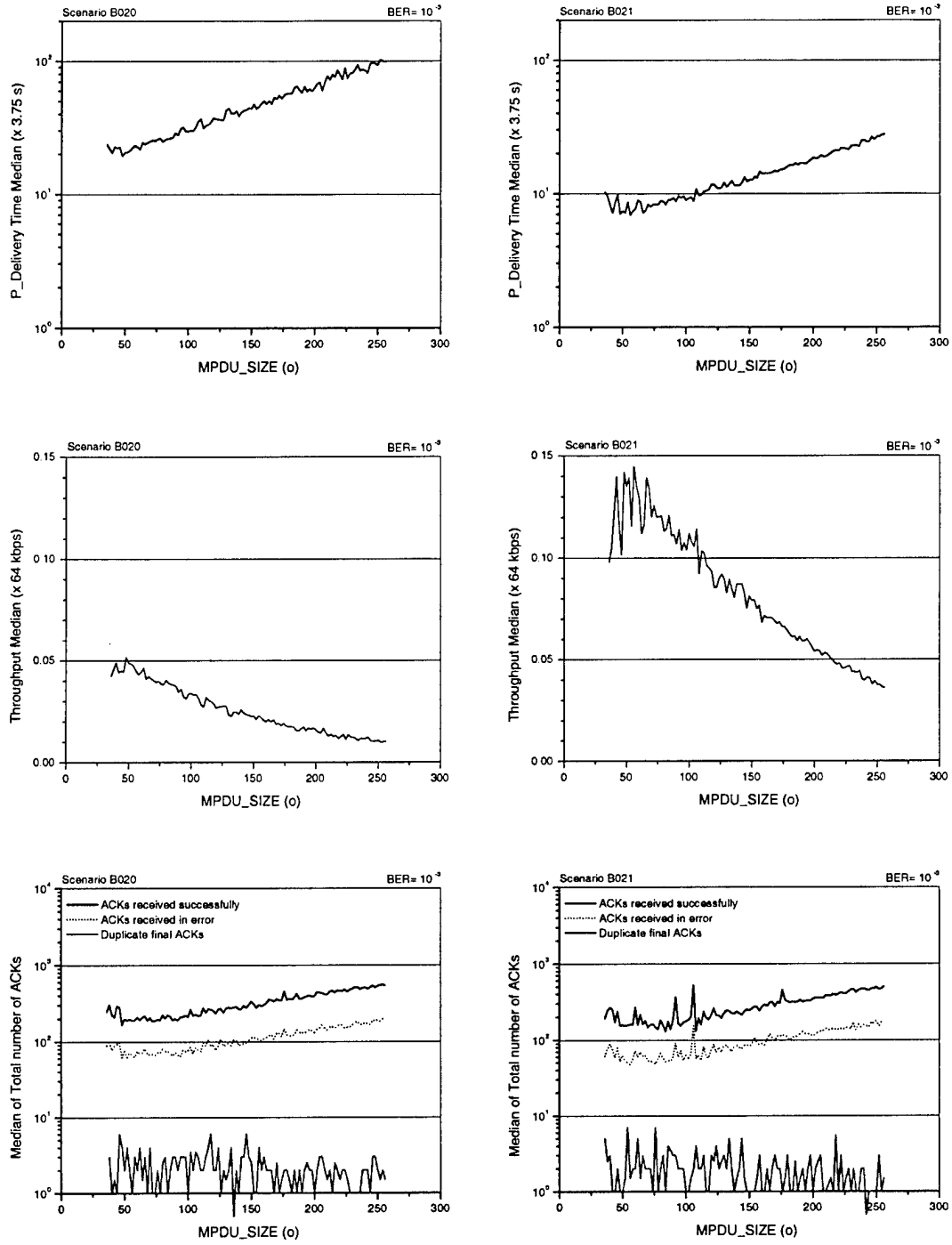


FIGURE 27. Performance for the lower MPDU_SIZE range (continued)

5.2.4.1 The Loss of Control Packets

While running scenarios where packets must be discarded because of errors, a number of problems were encountered and attributed to the way P_MUL handles some situations during a message transfer. This section identifies two of these problems.

1. Loss of the last ADDRESS_PDU by receivers

Having successfully received the final acknowledgement from the receivers, the sender returns an ADDRESS_PDU that does not contain the address of the receivers. At one or more receiving sites, that ADDRESS_PDU is discarded because of errors in the transmission. The receivers who have yet to receive acknowledgement from the sender that the latter has received their final acknowledgement keep sending their final ACK_PDU periodically at the rate commanded by the Min_Scan_Time timer. Having received all the required acknowledgments, the sender has disconnected. However, as far as those receivers are concerned, the message transfer is still active and will remain so until the message expiry time has been reached.

In the CRC model of P_MUL, a Max_Connection_Time_Out timer was included to prevent a receiver from expecting the completion of a message transfer that will never come. Otherwise, a receiver could stay active forever since some messages could possibly have an expiry time that is far beyond the time required to complete a message transfer. Receivers who were in EMCON mode when the message expired would stay active indefinitely if the final ADDRESS_PDU were to be received in error.

2. False ADDRESS_PDU detection

A receiver can mistakenly believe that it has received the final ADDRESS_PDU when in fact it has not. In one of the scenarios, the sender sent two ADDRESS_PDUs to the receiver. The first had the receiver's address in it but it was discarded because of errors during the transmission; the second was received by the receiver. Since the receiver's address was not in the second ADDRESS_PDU, the receiver concluded that it had successfully received its final acknowledgment while in fact it had not. According to Section 410 b.2 of the ACP, the receiver can release all information about this message and can discard the ADDRESS_PDU silently, which the receiver in the CRC model did. The receiver was never aware that it had missed an ADDRESS_PDU. The sender on the other hand will resend to this receiver the DATA_PDUs that it thinks remain to be acknowledged, i.e., all those PDUs reported missing prior to the time the receiver sent its final acknowledgment. When these DATA_PDUs arrive at the receiver, paragraph 412 a. of the ACP requires the receiver to determine whether the DATA_PDUs have already been received. However, the receiver has no way to tell whether any particular DATA_PDU has been received or not because it has released all information about this message.

In the CRC model of P_MUL, the (source, message id) pair received during a scenario can be tracked. It is unclear whether in the ACP this information should be managed by P_MUL or by the application making use of P_MUL. The receiver looks at the MSID and

concludes that this message has already been delivered to the application, and thus silently discards the packet. The sender is never informed and keeps retransmitting the missing DATA_PDUs until the message expires.

Part of the problem has to do with the fact that the ADDRESS_PDU packet definition reserves only two bits in the header to notify the sender of how many ADDRESS_PDUs there are in a given message transfer.

The most desirable way to solve the problem is probably to have more than two bits to identify the ADDRESS_PDUS. Note that the problem of the false final ADDRESS_PDU detection can be solved for the case where the sender is sending only two ADDRESS_PDUs, i.e., one first, one last. Having received the last but missed the first, it can be concluded that the first is missing. But if there are more than two, then the problem exists again because there is no way to know for sure how many more ADDRESS_PDUs to expect in between the first and final ADDRESS_PDU.

The message field has 32 bits, enough to distinctly identify over 4 billion messages per source. The standard does not clearly state whether it is P_MUL's task to manage such a database. The question of keeping a history of the received messages arises because once the information about a message is released, there is no way back unless a history is kept somewhere. In the CRC model, a record is kept of every message that has successfully been received during a simulation scenario. The model does not accept any further PDUs for those messages. It has been implemented this way to be able to decide whether or not an ADDRESS_PDU has been received *before* (i.e. in the past), as is required in Section 410 of the P_MUL draft, paragraph c. The problem is how far back in time does "before" mean? It seems logical that by examining the *Message_Sequence_Number* field in the ADDRESS_PDU, the receiver can determine if it has already received the message. However, the CRC model does not keep track of messages received before the start of the simulation scenario.

5.2.5 The M Missing DATA PDUs Parameter

The ACK Re-Transmission timer is the primary means by which P_MUL flow controls message transfers. In the presence of noise, reducing the retransmissions interval increases the number of retransmissions, which generally results in higher utilization levels or loading of the network and faster response time.

Previous sections have shown that the network utilization level rarely exceeds 85% of the destination network capacity. Whether this is desirable or not, one can ask: can this limit be pushed further up? One obvious way to do so would be to reduce to a minimum the random delay controlling the sending of the ACK_PDUs. One other and more intriguing possibility is to increase the number of requests made to the sender when receivers are missing PDUs. This can be done easily by adjusting the value of the M_Missing_DATA_PDUS parameter. Its value controls the number of missing DATA_PDUs a receiver can request from the sender when sending one ACK_PDU. If the receiver is

missing more than M data packets then it must request the missing packets by sending several requests.

In the ACP 142, it is said that the Length_of_ACK_Info_Entry — which in the ACK_PDU packet is the field controlling the maximum number of missing PDUs to be listed — is adaptable in length to the quality of the transmission. The ACP does not recommend nor provide any indication on how to adjust the field for best results.

The size of an ACK_PDU is:

$$\text{ACK\_PDU Size} = 24 + 2M \tag{10}$$

where M is the maximum number of missing DATA_PDUs listed in the ACK_PDU packet. Since an ACK_PDU cannot exceed the size of MPDU_SIZE octets, then

$$M \leq \frac{(\text{MDPU\_SIZE} - 24)}{2}. \tag{11}$$

Given that a minimum size P_MUL packet has 36 octets, the maximum number of M missing DATA_PDUs that can be listed in a minimum size P_MUL packet is 6 PDUs. For an MPDU_SIZE of 64, M equals 20 PDUs. This last result is used in the next two scenarios to see the effect of adjusting the M_Missing_DATA_PDUS parameter over a range of 1 to 20, in increments of 1 PDU. The MPDU_SIZE parameter is set at 64 octets as this value was shown in Section 5.2.4 to significantly improve the performance over those obtained in Section 5.2.3 for packets having a size of 1024 octets. Parameters for both scenarios, D022 and D024, are summarised in Table 19.

Table 19. Main parameter settings for scenarios D022 and D024

	Parameter	Default Value	Scenario[1]	
			D022	D024
P_MUL Core	ACK_Re-Transmission_Min		6.0 s	0.5 s
	MPDU_SIZE		64 o	64 o
	Delete_DATA_PDUS_Time	60 s		
	M_Missing_DATA_PDUS	8 pdus	*variable*	*variable*
	ACK_PDU_Jitter	1.0 s		
	Min_Scan_Time	60 s		
Source Subnet	Message Expiry Time	4 hr		
Dest. Subnet	Destination BER		$5\ 10^{-3}$	$5\ 10^{-3}$

1. Values in italic overwrite default values shown in Table 15.

Simulation results are shown in Figures 28 and 29. Note that the bit error rate has been increased by a factor of 5 over the test conditions of Section 5.2.4. The graphs in this section show the results of several simulation runs, each with a different random number seed, for each set of parameter values. The vertical lines shown indicate the minimum and maximum results obtained for a particular set of parameter values. The curve shown is created by linking the median values of each set of runs. This format is used in the remainder of this document for graphs showing the results of runs with multiple seed values.

In both simulations, a small M_Missing_DATA_PDUS value (M<6) caused too many retransmissions and excess ACK_PDUs, resulting in a reduction of throughput and an increase in P_Delivery Time. For larger M_Missing_DATA_PDUS values (5<M<21), the performance is nearly constant. There is perhaps a slight increase in bus utilization and packet ratio as M increases, but the high variance in the results makes this unclear.

Although all runs are successful, a fairly large proportion of the runs (about 1 in 30) remain active until the message expires. These occurrences are caused by the loss of ADDRESS_PDU control packets.

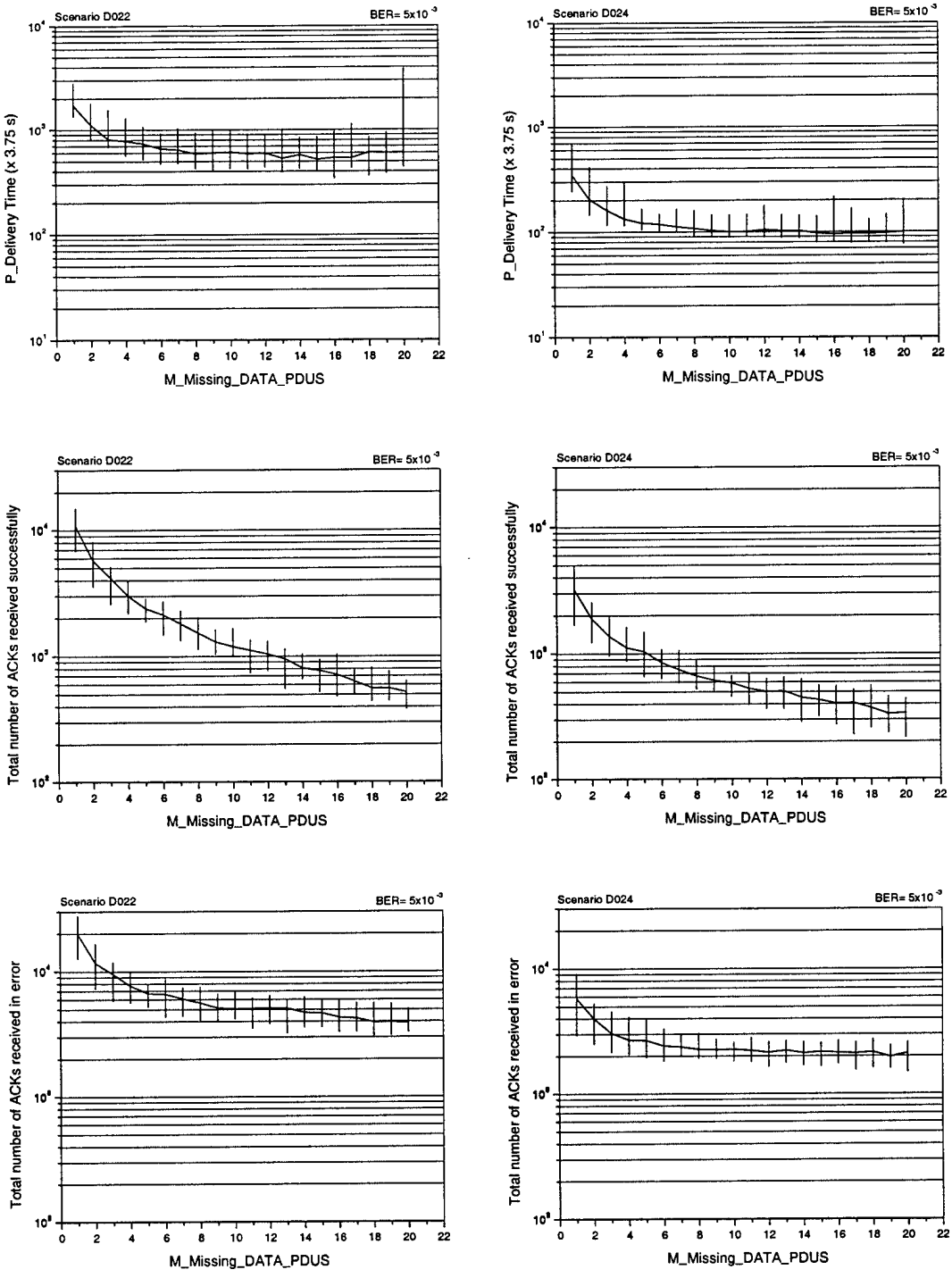


FIGURE 28. Effect of the M_Missing_DATA_PDUS parameter on performance

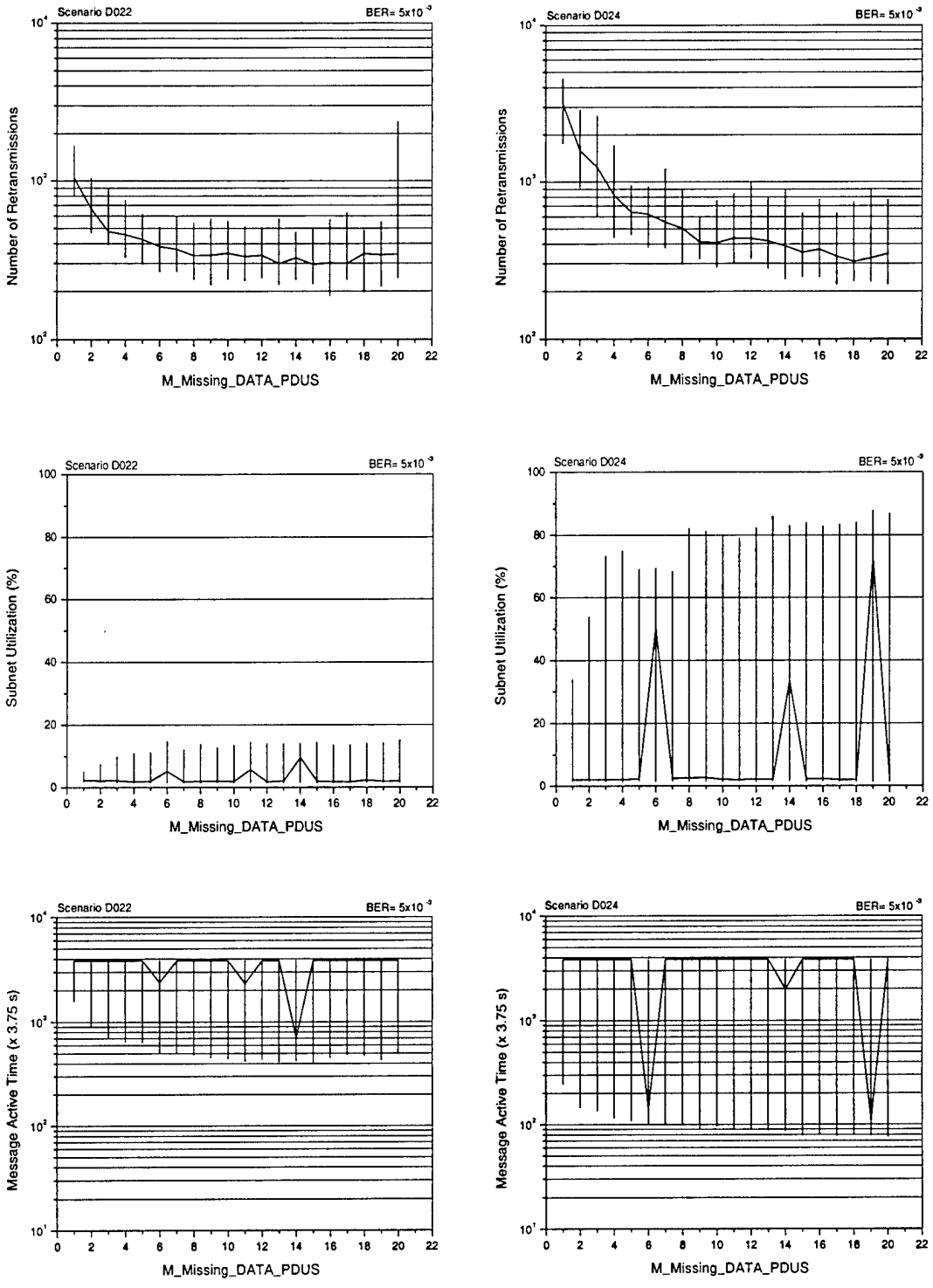


FIGURE 29. Effect of the M_Missing_DATA_PDUS parameter on performance (continued)

5.2.6 Coding

In all previous scenarios, no error correction mechanism was used. As has been mentioned, an erasure coding scheme was added to the CRC model to investigate if the robustness of message delivery in lossy conditions could be improved. In order to analyse the effect of erasure coding on message transmission two simulations, D030 and D040, differing only in their erasure coding parameters, were run. Both simulations consisted of ten runs at each of twelve different BER settings. The main parameters for scenarios D030 and D040 are summarised in Table 20.

In scenario D030 no erasure coding was used. In scenario D040 a $(3k/2,k)$ erasure code was used, where k can be expressed as:

$$k = \frac{\text{Message Length}}{\text{Block Size}} \tag{12}$$

Since erasure code blocks in the model are always constrained to the maximum size that may be fit into a PDU of size MPDU_SIZE (see Section 3.2.2), the block size may be calculated using:

$$\text{Block Size} = \text{MPDU\_SIZE} - \text{DATA\_PDU\_HEADER\_SIZE} \tag{13}$$

In scenarios D030 and D040 MPDU_SIZE is set to 64 octets. Given that the DATA_PDU header size is 16 octets, the resulting block size is 48 octets. Therefore, for the 30 000 octet message size specified in D040 k is equal to 30 000 divided by 48, or 685, and n is $3k/2$, or 938. This gives erasure coding parameters of (938,625) for scenario D040.

The results for scenarios D030 and D040 are shown in Figures 30 and 31. Without any error correction mechanism, the bus utilization decreases significantly with increasing bit error rate. Use of erasure coding increases bus utilization at lower bit error rates (1.8×10^{-3} or less), but fails to produce much effect at higher bit error rates. In both simulations, the bus utilization occasionally drops to near zero because of the loss of significant control packets.The use of erasure coding also improves the throughput and reduces the number of retransmissions at lower bit error rates. However, beneficial effects disappear at higher bit error rates.

A large number of ACK_PDUs are produced, even at lower bit error rates, in simulation D030. The use of erasure coding reduces the number of ACK_PDUs received in error, but increases the total number of ACK_PDUs received, a large proportion of these being duplicate final ACK_PDUs. Since final ACK_PDUs are smaller than ACK_PDUs listing missed PDUs the larger proportion of duplicate final ACK_PDUs seems to be the cause of the reduced number of erroneously received ACK_PDUs when using erasure coding.

Erasure coding provides redundancy in the data sent. The amount of redundancy is dependant on the parameters used. As n grows in proportion to k more redundancy is

introduced. Does increasing redundancy improve performance? In order to investigate this possibility two further scenarios using different erasure coding parameters were run. Both were identical to scenario D040 except in their erasure coding parameters. The first, scenario D042, used erasure coding parameters of ($7k/4,k$), i.e. (1094,625). The second, scenario D044, used erasure coding parameters of ($2k,k$), i.e. (1250,625). Selected results from all four erasure coding scenarios are shown in Figure 32.

The results of the simulations show no significant difference in the performance of P_MUL under varied erasure coding parameters. All of the scenarios using erasure coding show improvement over the scenario where no erasure coding was used for low bit error rates. However, for all coding parameters the improvement for high bit error rates is minimal.

Table 20. Main parameter settings for scenarios D030 and D040

	Parameter	Default Value	Scenario[1]	
			D030	D040
P_MUL Core	ACK_Re-Transmission_Min		0.5 s	0.5 s
	MPDU_SIZE		64 o	64 o
	Delete_DATA_PDUS_Time	60 s		
	M_Missing_DATA_PDUS	8 pdus		
	ACK_PDU_Jitter	1.0 s		
	Min_Scan_Time	60 s		
FEC	Erasure Code (n,k)		none	(3k/2,k)
	Erasure Block Size	MPDU_SIZE-16	48 o	48 o
Source Subnet	Message Expiry Time	4 hr		
Dest. Subnet	Destination BER		*variable*	*variable*

1. Values in italic overwrite default values shown in Table 15.

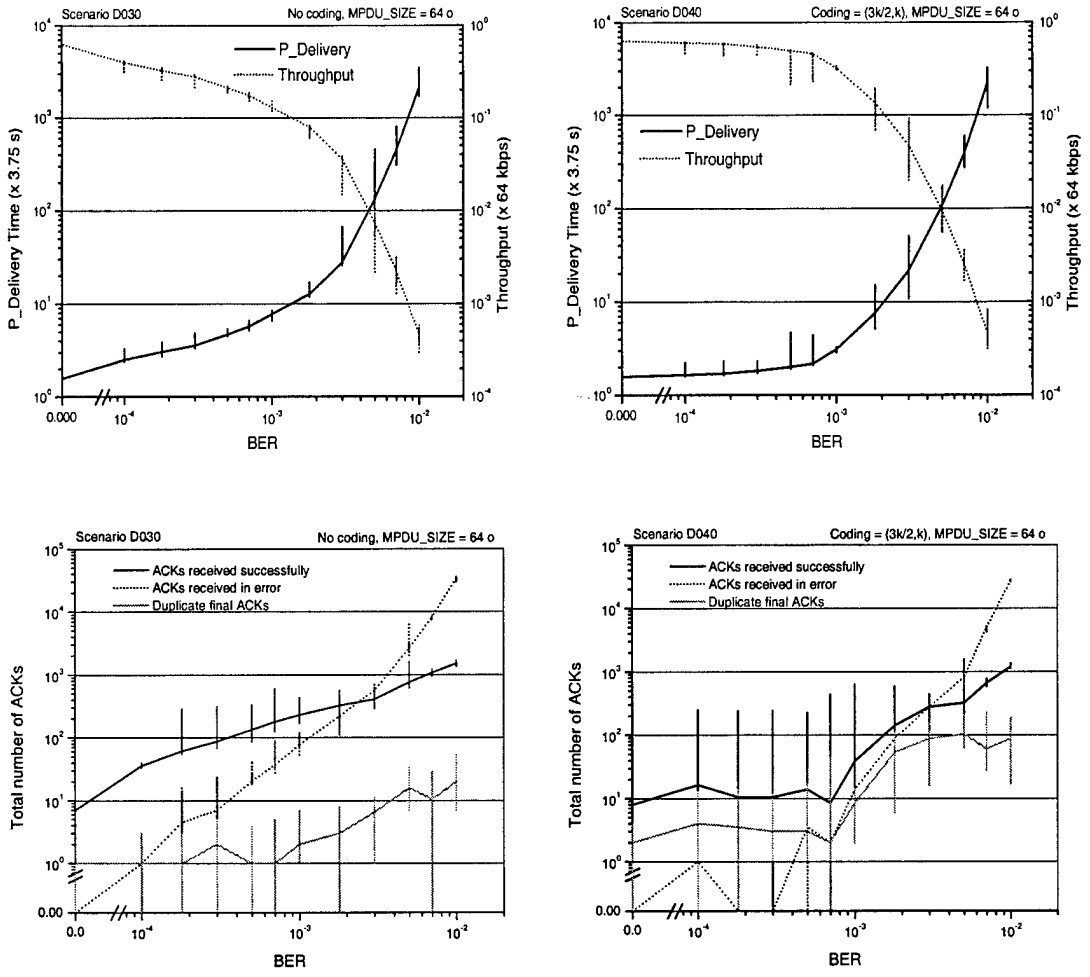


FIGURE 30. Use of erasure block coding to improve performance

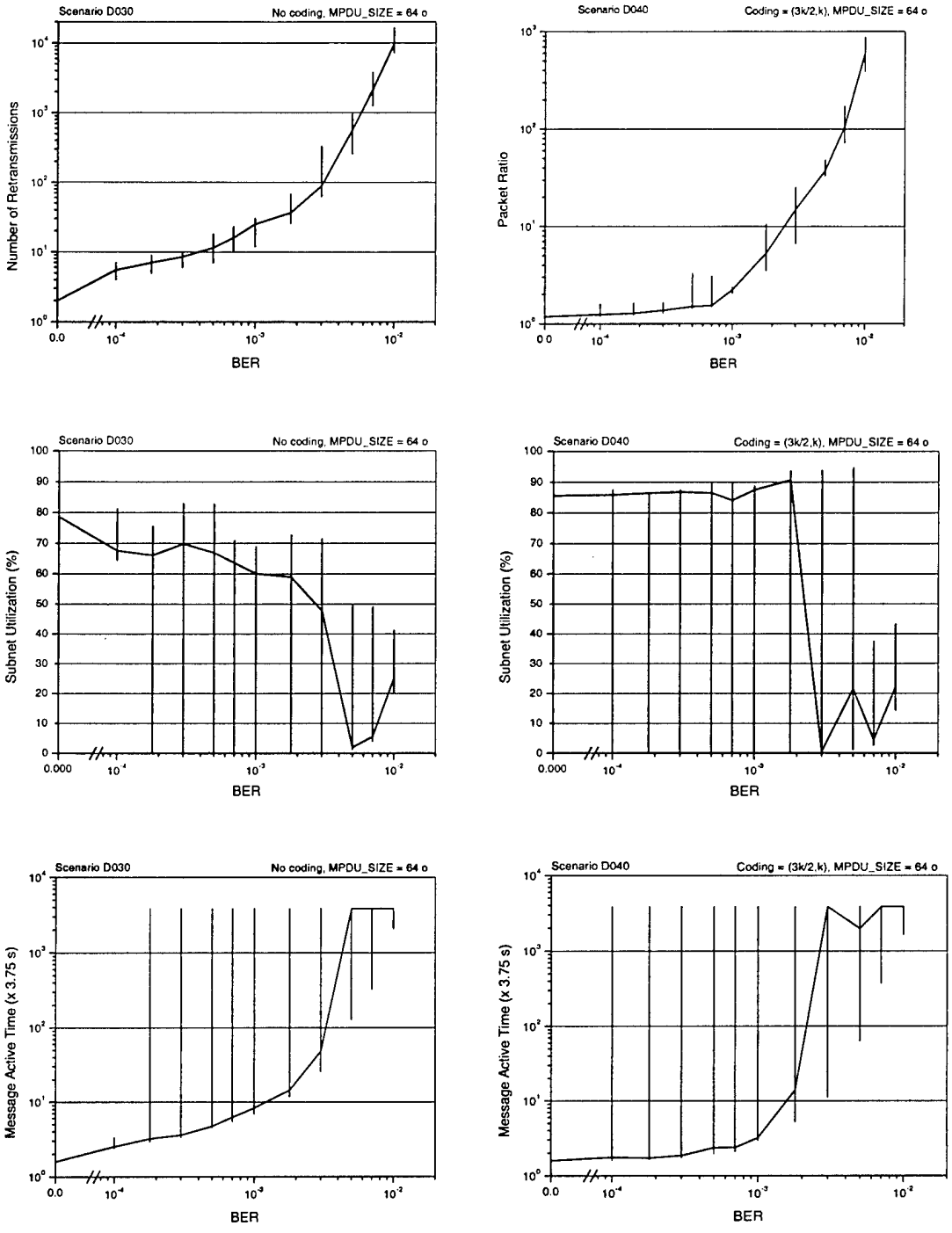


FIGURE 31. Use of erasure block coding to improve performance (continued)

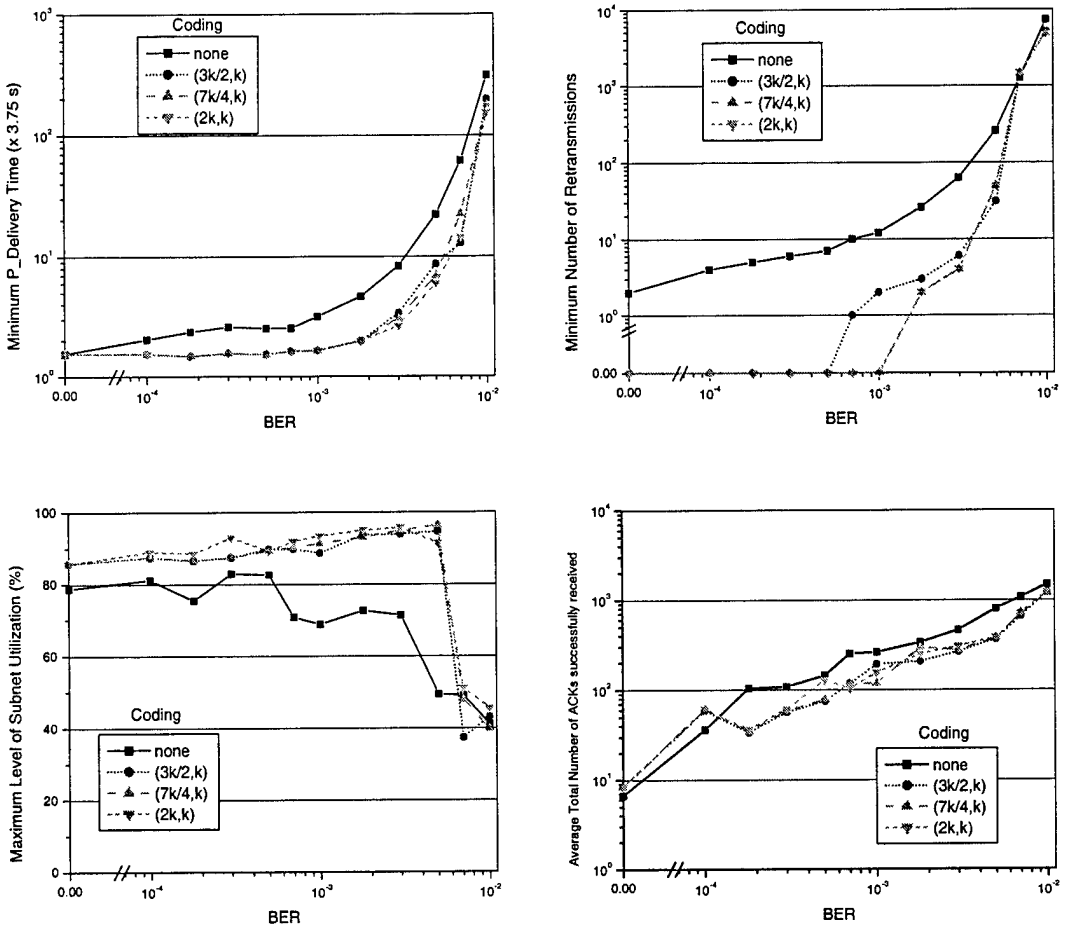


FIGURE 32. Impact of erasure coding parameters on performance

As was previously shown, the size of the PDUs has a large impact on P_MUL's performance. It is interesting to see if erasure coding is able to counteract the detrimental effects of larger PDU sizes. Scenarios D035 and D045 explore the use of erasure coding with larger packet sizes. The main parameters used in scenarios D035 and D045 are shown in Table 21. Scenario D035 is identical to scenario D030 except that the MPDU_SIZE parameter is 256 octets. Similarly, scenario D045 is identical to scenario D040 except for the MPDU_SIZE parameter, which is 256 octets.

Given that the DATA_PDU header size is 16 octets, the resulting block size is 240 octets. Therefore, for the 30 000 octet message size specified in D045, k is equal to 30 000 divided by 240, or 125, and n is $3k/2$, or 188. This gives erasure coding parameters of (188,125) for scenario D045.

Simulation results for scenarios D035 and D045 are shown in Figure 33 and Figure 34, respectively. Here again, gains due to the use of erasure coding are evident at lower bit error rates. However, the use of erasure coding is not sufficient to save message transmission at higher bit error rates. In both scenario D035 and scenario D045 the sender fails to deliver the message successfully at bit error rates higher than 0.003. Note that at low bit error rates scenarios D035 and D045 have lower P_Delivery times and higher throughput than scenarios D030 and D040, due to the increased efficiency of using larger packet sizes.

Table 21. Main parameter settings for scenarios D035 and D045

	Parameter	Default Value	Scenario[1]	
			D030	D040
P_MUL Core	ACK_Re-Transmission_Min		0.5 s	0.5 s
	MPDU_SIZE		256 o	256 o
	Delete_DATA_PDUS_Time	60 s		
	M_Missing_DATA_PDUS	8 pdus		
	ACK_PDU_Jitter	1.0 s		
	Min_Scan_Time	60 s		
FEC	Erasure Code (n,k)		none	(3k/2,k)
	Erasure Block Size	MPDU_SIZE-16	240 o	240 o
Source Subnet	Message Expiry Time	4 hr		
Dest. Subnet	Destination BER		*variable*	*variable*

1. Values in italic overwrite default values shown in Table 15.

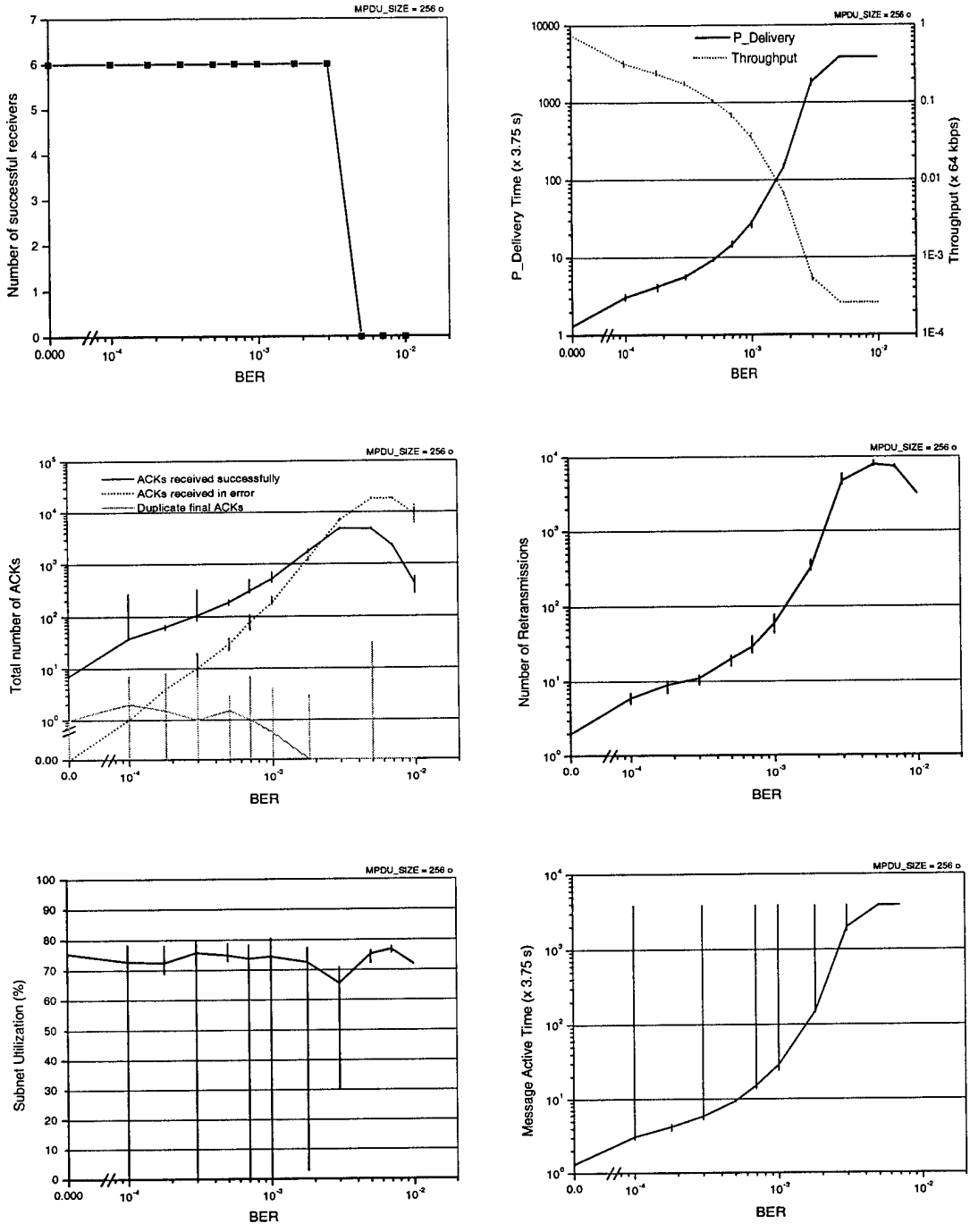


FIGURE 33. Use of erasure coding with large MPDU_SIZE

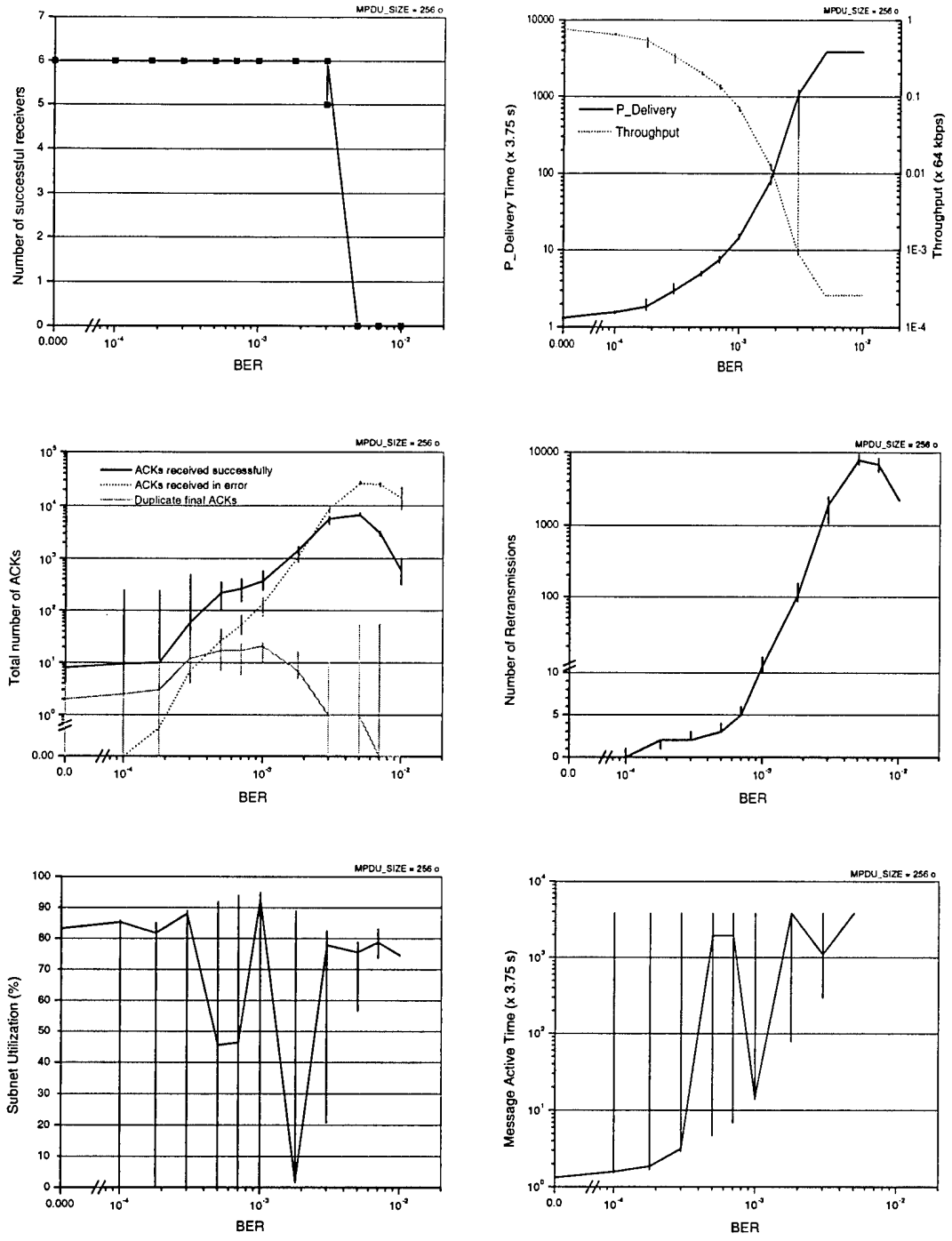


FIGURE 34. Use of erasure coding with large MPDU_SIZE (continued)

5.2.7 Message Length

In the Internet-Draft it is suggested that P_MUL may be used for "Multicast File Distribution". Multicast file distribution is a mechanism through which one user can distribute a file simultaneously to a group of receivers. The types of files likely to be distributed in this fashion are large (examples given in the Internet-Draft are software packages and weather maps). In order to investigate the impact of very large message sizes on P_MUL's performance two scenarios, D050 and D055, were created.

The main parameters for scenarios D050 and D055 are outlined in Table 22. Both D050 and D055 are identical to scenario D030 except in the length of the message to be transferred. The message length for scenario D030 was 30 ko, while the message lengths for D050 and D055 are 300 ko and 3 Mo respectively. Each scenario was run with eleven different bit error rate values, with each bit error rate / message length combination being run ten times, each time with a different random number seed.

Table 22. Main parameter settings for scenarios D050 and D055

	Parameter	Default Value	Scenario[1]	
			D050	D055
P_MUL Core	ACK_Re-Transmission_Min		0.5 s	0.5 s
	MPDU_SIZE		64 o	64 o
	Delete_DATA_PDUS_Time	60 s		
	M_Missing_DATA_PDUS	8 pdus		
	ACK_PDU_Jitter	1.0 s		
	Min_Scan_Time	60 s		
Source Subnet	Message Expiry Time	4 hr		
	Message Length	30 ko	*300 ko*	*3 Mo*
Dest. Subnet	Destination BER		*variable*	*variable*

1. Values in italic overwrite default values shown in Table 15.

It is to be expected that message delivery time will increase as the message length increases. Therefore, in order to fairly compare scenarios D030, D050, and D055 it is necessary to normalise the results with respect to the message length. The normalised P_Delivery time for a message may be calculated as:

$$\text{Normalised P\_Delivery Time} = \frac{\text{P\_Delivery Time}}{\text{Message Length/Nominal Subnetwork Capacity}}$$

Selected results from scenarios D050 and D055 are shown in Figures 35 and 36. Once the P_Delivery times have been normalised, they are similar up until the point at which the message transfer begins to fail. Message transfer fails at high bit error rates in both scenarios because the message expiry time is reached after four hours without all receivers having successfully acknowledged the message. Message failure occurs at lower

bit error rates as the message size becomes larger, which is expected as larger messages should take a proportionally longer amount of time to deliver. Should the expiry timer have been increased with increasing message size, it is likely that the normalised results obtained at higher bit error rates would have continued to be similar.

Bus utilization in both scenarios is higher than that achieved in scenario D030 with a message size of 30 ko. Furthermore, the utilization achieved when transferring a message of 3 Mo is higher than that obtained when transferring a message of 300 ko – approaching 100% at low bit error rates. Bus utilization decreases as the bit error rate increases up until the point at which message transfer begins to fail. When message transfer begins to fail utilization increases sharply and the number of retransmission decreases correspondingly. The low number of retransmissions suggests that at higher bit error rates the sender never receives an ACK from one or more receivers and is therefore retransmitting the entire message numerous times, saturating the network.

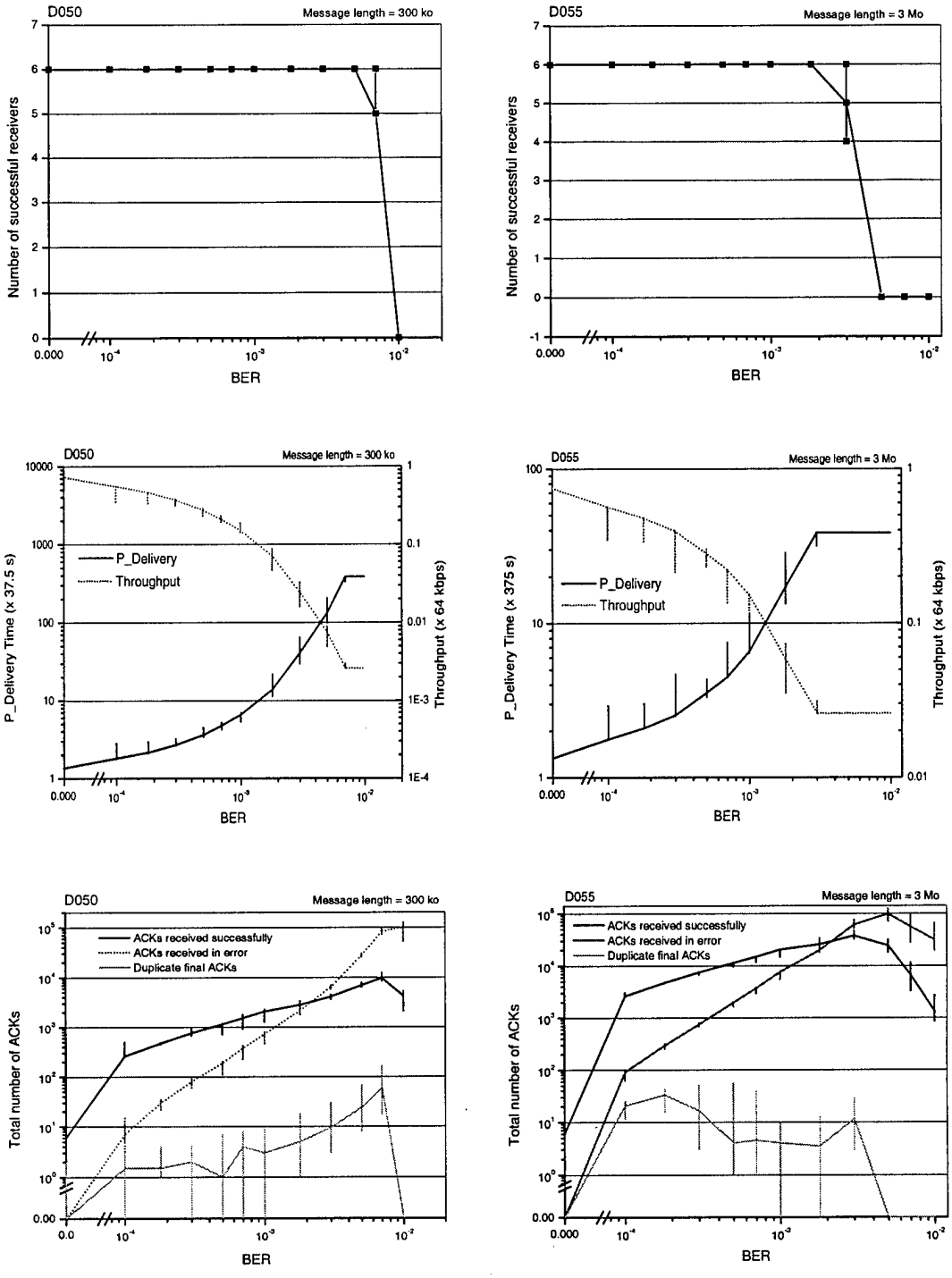


FIGURE 35. Impact of message size on performance

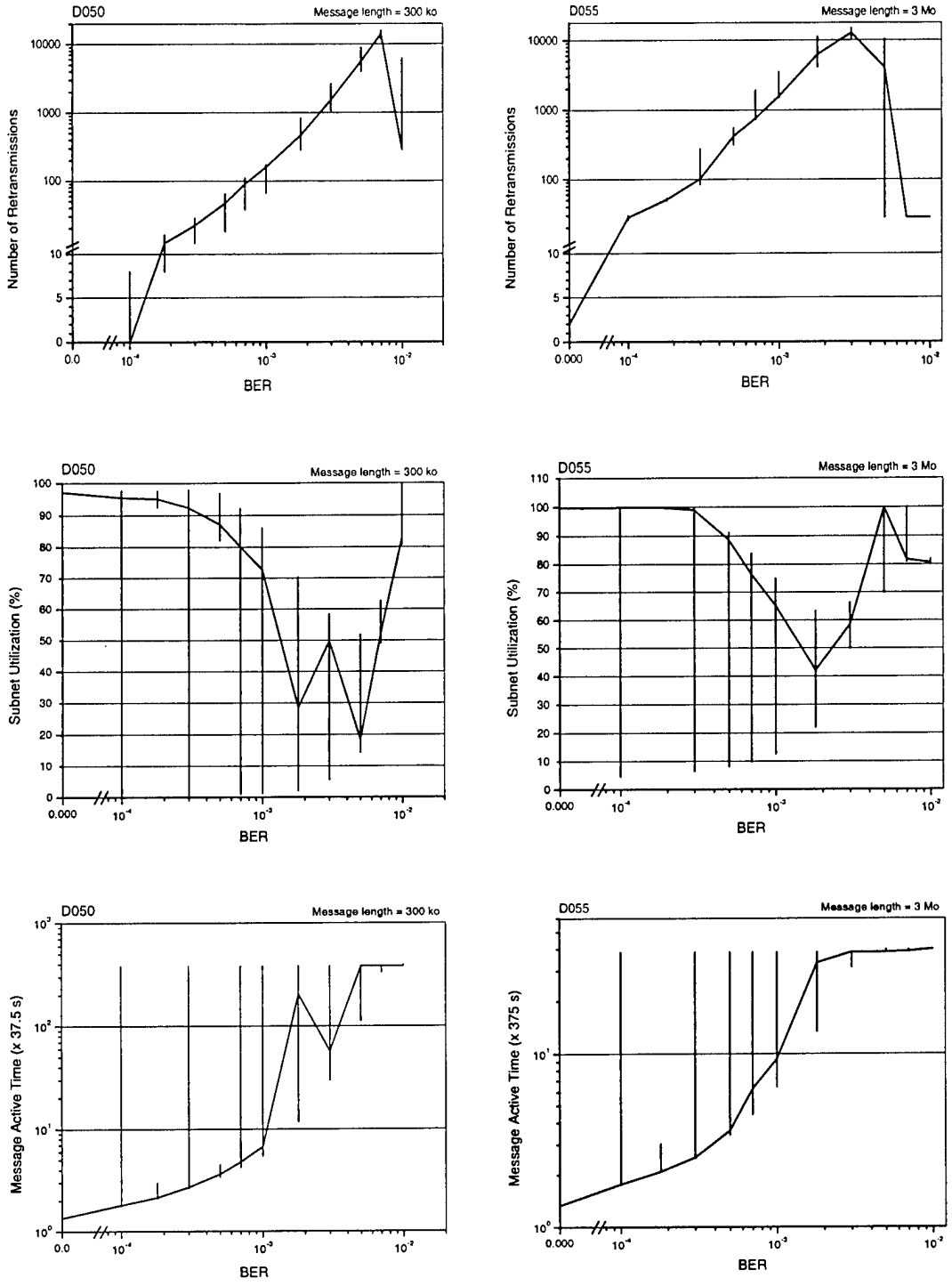


FIGURE 36. Impact of message size on performance (continued)

A comparison of the normalised P_Delivery Time achieved for the three different message lengths is shown in the left-hand graph in Figure 37. The normalised P_Delivery time is nearly independent of the message length. The curves diverge at higher bit error rate values because the message transfers aborted when the expiry time was reached. If we assume that the normalised P_Delivery Time would have continued to be independent of the message length at higher bit error rates had the messages not expired, we can extract from the normalised P_Delivery Time the expected maximum message length that could be transferred in a given period of time. The maximum possible message length can be expressed as:

$$\text{Max. Message Length} = \frac{\text{Maximum Transfer Duration}}{\text{Normalised P\_Delivery}/\text{Nominal Subnetwork Capacity}} \quad (14)$$

The two curves in the right-hand graph of Figure 37 show the maximum message length that may be transferred in a one hour and a four hour period. For both curves, the nominal subnetwork capacity is taken to be 64 kbps. In conclusion, it would appear that when time is a constraint, P_MUL is rather limited in its capability of transferring large files when the transfer must be done in a noisy environment.

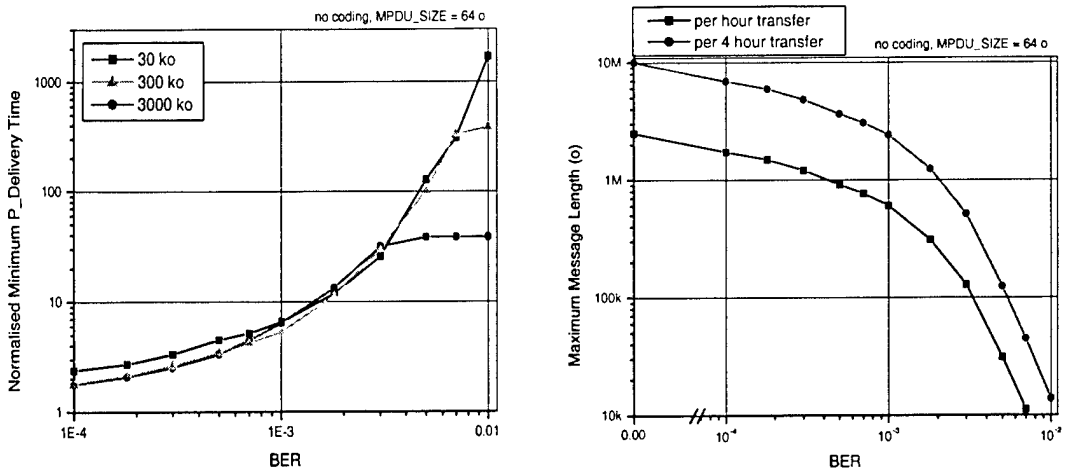


FIGURE 37. Impact of message size on performance (continued)

5.2.8 Number of receivers

In all of the scenarios seen thus far, the number of message receivers has been small – 6 or 12 receivers on a single subnet. According to the ACP 142, P_MUL was designed to be "deployed in a network environment consisting of hundreds of nodes, but less than one thousand". Is P_MUL capable of scaling to accommodate large receiver group sizes without an unacceptable degradation in performance?

The simulation model developed at CRC can support topologies consisting of hundreds of nodes and is only limited in size by the available computer memory. Of course, the time required for the simulation of very large networks can become impractical, especially when message transfers involve large message sizes. Because of the numerous problems encountered during the development phase of the P_MUL model, no scenarios with message receiver group sizes of hundreds of nodes were studied. Nevertheless, scenario D060 is a last minute attempt to explore the effects of increasing receiver group size on P_MUL. The size of the group of recipients is increased from 6 to 42 in increments of 6 receivers. Scenario D060 is identical to scenario D030, except in its network topology and multicast group membership. The main parameters of scenario D060 are outlined in Table 23.

Table 23. Main parameter settings for scenario D060

	Parameter	Default Value	Scenario
			D060
P_MUL Core	ACK_Re-Transmission_Min		0.5 s
	MPDU_SIZE		64 o
	Delete_DATA_PDUS_Time	60 s	
	M_Missing_DATA_PDUS	8 pdus	
	ACK_PDU_Jitter	1.0 s	
	Min_Scan_Time	60 s	
Network Topology	Multicast Group (Address:Members)		*variable*
	Group-size / Subnet-size		*variable*
Source Subnet	Message Expiry Time	4 hr	
	Message Length	30 ko	
Dest. Subnet	Destination BER		*variable*

The topology is composed of one T-node (sender), attached by one high-speed (10 Mbps) zero delay (0 ms) link to a transit router. Seven high-speed (10 Mbps) zero delay (0 ms) links connect the transit router to seven separate destination broadcast networks (subnet_0 to subnet_6). Each broadcast network consists of one 64 kbps bus to which is attached a group of six recipients (node_1 to node_6). From this topology, seven different multicast groups are created. The first multicast group contains all of the recipi-

ents in the first subnetwork, the second all of the recipients in the first and second subnetworks and so on. The sub-network level topology for scenario D060 is shown in Figure 38.

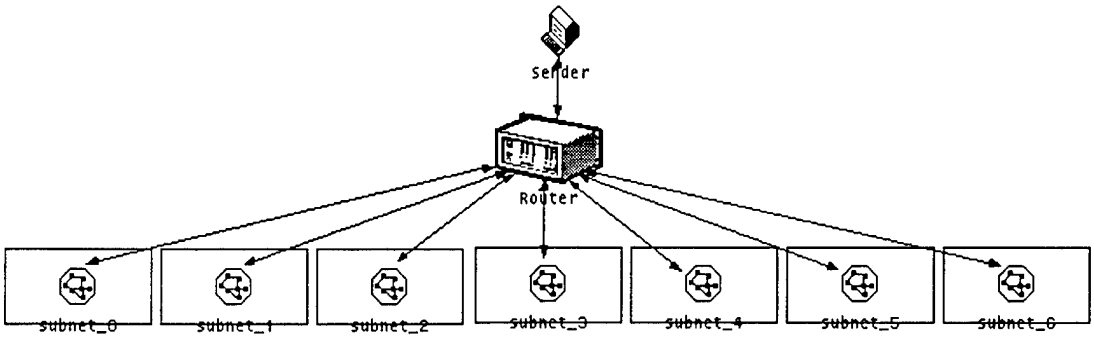


FIGURE 38. Topology for scenario D060

Selected results for scenario D060 are shown in Figure 39. P_MUL appears to react well to increased receiver group size. As the number of receivers increases the P_Delivery Time and throughput remain fairly constant. As would be expected, the number of ACK_PDUs received increases with increasing reciever group size. The number of ACK_PDUs received for a subset of bit error rates were normalised with respect to the number of ACK_PDUs received for the smallest group size. The resulting graph demonstrates that the number of ACK_PDUs received increases fairly linearly with increasing group size.

With increasing group size there is a larger proportion of runs which fail due to the loss of significant control packets. With a multicast group size of six just under a quarter of all runs had at least one receiver which missed a significant control packet. When the group size reached forty-two half of all runs experienced this problem. This is as expected, for as the group size increases the number of ADDRESS_PDUs increases. Furthermore, the proportion of vulnerable "middle" ADDRESS_PDUs increases.

The results obtained with group sizes up to forty-two show a pattern that may well extend to larger group sizes. The results suggest that group size does not impact significantly the P_Delivery Time or throughput achievable by P_MUL. However, with increasing group size ACK implosion and the loss of significant control packets could become problematic issues.

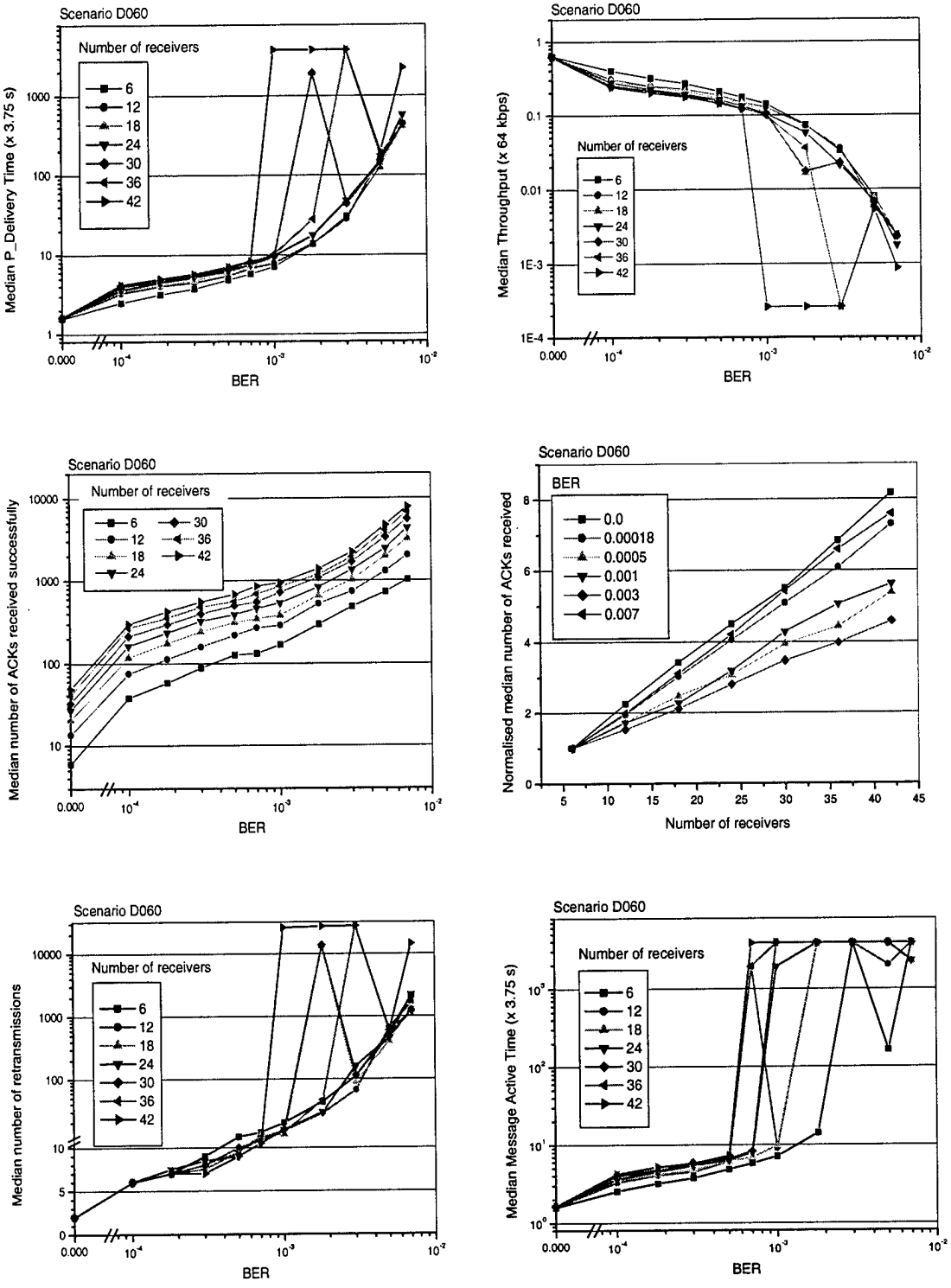


FIGURE 39. Effect of number of receivers on performance

5.2.9 Multicast Group Address Rejection

P_MUL uses a *Silent Procedure* to set up multicast groups. Once a group address has been selected, the address is made globally known by sending a REQUEST_PDU to each possible transmitter on the network. After waiting for a predefined time interval, the node requesting ownership of the address will assume that the address is legitimate if no transmitter in the network has rejected its request by returning a REJECT_PDU. However, it is possible that under noisy channel conditions, that either the REQUEST_PDU or the REJECT_PDU, or both, be discarded because of bit error. In this case, the use of the requested multicast address will result in some additional overall network load because of accidental packet duplications for nodes sharing the same multicast address.

In Section 302.e of the ACP, it is pointed out that if all T-nodes track the ownership of multicast group addresses, then it would be expected that a multicast group rejection would rarely be required. This should certainly be true for nodes that have been on-line for some time. For those other cases where the request rejection procedure might still be required it is worth asking how well the proposed scheme performs, especially in a noisy channel environment.

The probability that no errors occur in a packet of size N corresponds to N successive events with probability (1-*ber*) as described by (8) in Section 5.2.1. Since both the REQUEST_PDU and the REJECT_PDU must be transferred with no error to notify the transmitter of any rejection, the probability $P_{0,k}$ of having a correct notification from a T-node k is:

$$P_{0,k} = (1-ber_k)^{N_{Req}}(1-ber_k)^{N_{Rej}} = (1-ber_k)^{N_{Req}+N_{Re}} \tag{15}$$

where ber_k is the average bit-error-rate observed between the sender and the receiver k, N_{Req} and N_{Rej} are the sizes in bits of the REQUEST_PDU and REJECT_PDU respectively. The probability $P_{e,k}$ that the notification fails because of one or more errors present in the two packets is:

$$P_{e,k} = 1 - P_{0,k} \tag{16}$$

The procedure requires that the REQUEST_PDU be sent to all T-nodes in the network. If we assume that the network is composed of $T+1$ independent T-nodes (the sender is itself a T-node), sharing a portion of the spectrum where the bit-error-rate is the same for all links, i.e. $ber_1=ber_2=...=ber_k=...=ber_{T-1}=ber_T$, the chances of correctly notifying the transmitter that the multicast address requested is in use will depend on the probability of receiving at least one error-free REJECT_PDU. Or stated differently, the transmitter will be correctly notified in all instances except those cases where none of the nodes succeeds in transferring an error-free REJECT_PDU. This latter condition can easily be evaluated. If $P_{e,T}$ designates the probability that none of the T nodes returns an error-free REJECT_PDU then:

$$P_{e,T} = P_{e,1}P_{e,2}\ldots P_{e,T} = P_{e,k}^{T} = (1 - P_{0,k})^{T} \quad (17)$$

Therefore, the probability $P_{0,T}$ of being correctly notified is:

$$P_{0,T} = 1 - (1 - P_{e,k})^{T} \quad (18)$$

Inserting (15) in (18):

$$P_{0,T} = 1 - (1 - (1 - ber_k)^{N_{Req} + N_{Rej}})^{T} \quad (19)$$

Given that the sizes of the REQUEST_PDU and the REJECT_PDU are the same and is equal to 20 octets or 160 bits, expression (19) is evaluated numerically over a wide bit-error-rate range to produce the curves shown in Figure 40. The scheme appears to be relatively robust. For instance, in a small network composed of 7+1 T-nodes, only 1 address in 10 000 requests would end up being reused if the bit-error-rate observed over the links was not to exceed one error in 1000 bits. A similar evaluation can be performed with P_MUL attached to a transport and network layers. Figure 41 provides performance data when the UDP, IP and subnetwork overhead are included in the total packet size. N_{Req} and N_{Rej} were set to 56 octets each, the bit fields being allocated as follows: 20 octets for the P_MUL PDU, 8 octets for the UDP header, 20 octets for the IP header and 8 octets for the subnetwork frame. For the same small network composed of 7+1 T-nodes where the bit-error-rate observed over the links does not exceed one error in 1000 bits, about 270 addresses in 10 000 requests (2.7%) would end up being reused.

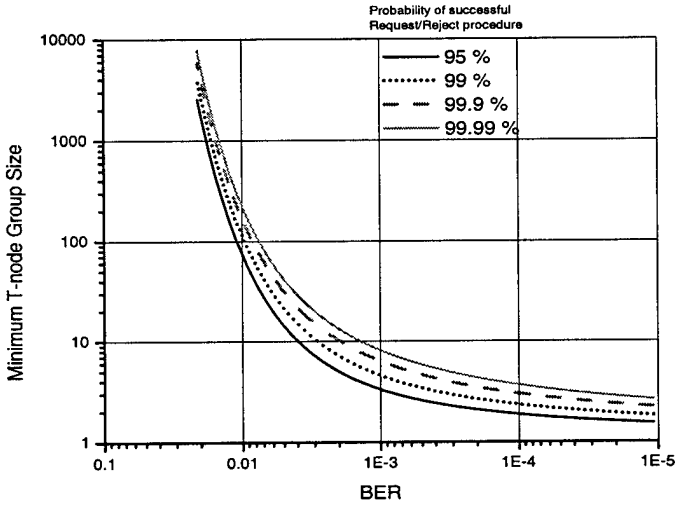


FIGURE 40. Probability of successful rejection of a duplicate address (no overhead)

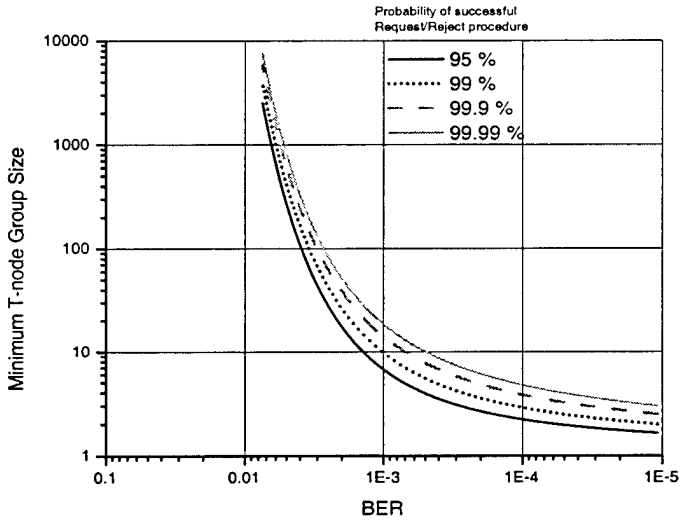


FIGURE 41. Probability of successful rejection of a duplicate address (with added overhead)

In order to validate the CRC model with respect to the theoretical performance of the dynamic address negotiation process a scenario, D070, was constructed. The CRC model of P_MUL allows the user to request that a reserved "*in use*" address be the first chosen by the address negotiation process. This reserved address is always considered to be in use, meaning that any T-node who receives a REQUEST_PDU for this address will automatically respond with a REJECT_PDU. In each run of scenario D070 one hundred messages are requested to be sent, each one of them requiring a dynamically allocated address. For each message, it is requested that the first address chosen be the "*in use*" address. Therefore, in ideal (error-free) conditions scenario D070 would request the "*in use*" address one hundred times and reject this address one hundred times.

Scenario D070 examines the likelihood of address negotiation failure as the bit error rate of the network increases. The values for the bit error rate of the network are scanned between 0.0025 and 0.0075 in increments of 0.0005 error per bit. For each bit error rate setting ten simulation runs, each with a different seed value, were run. The results from scenario D070 are shown in Figure 42. This graph shows the probability of successful address negotiation for varying bit error rates. The address negotiation mechanism is considered to have failed if the "*in use*" address is not rejected. The performance metric *Number of addresses rejected* is used to determine how many of the address negotiation attempts failed. If all attempts were successful, the Number of addresses rejected would be one hundred or 100%. Every failed negotiation attempt will reduce the Number of addresses rejected by one or 1%. The smooth curve shown in Figure 42 is the theoretical

probability of success for the negotiation process. The results of scenario D070 indicate that P_MUL's performance is very close to the theoretical prediction.

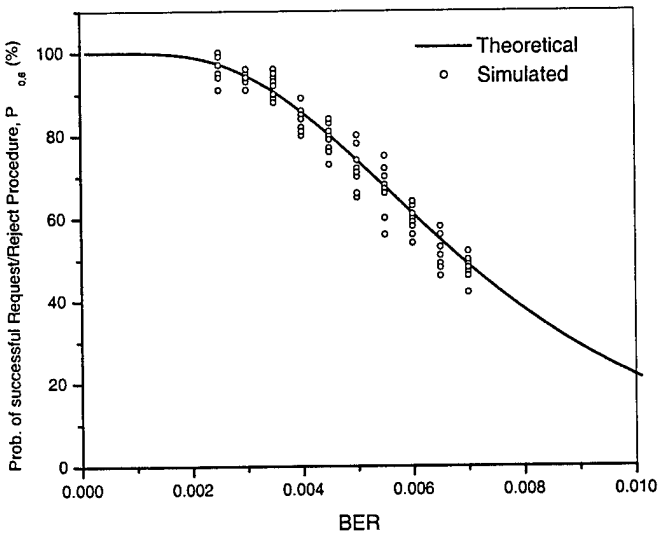


FIGURE 42. Theoretical and simulated performance of dynamic address negotiation

5.2.10 Multicast Address Pool Size (MAPS)

The results derived in the previous section and shown in Figures 40 and 41 can be used to determine the size of the multicast address pool (MAPS) required to detect with a certain level of probability $P_{\bar{u},T}$ the non-uniqueness in the ownership of the requested multicast group addresses. If V_c designates the expected volume of concurrent messages in the network for which dynamic multicast group addresses are necessary, then:

$$P_{\bar{u},T} = \left(\frac{V_c}{\text{MAPS}}\right) P_{e,T} \tag{20}$$

The ratio in brackets represents the probability that a node that is about to setup a new multicast group randomly selects a group address that is already in use in the network. Obviously, it is assumed that the node issuing the request has no knowledge of the existence of the V_c group addresses, otherwise the node would not select any one of them. As in the previous section, $P_{e,T}$ designates the probability of unknowingly reusing a group address when none of the T nodes manage to return an error-free REJECT PDU. Rearranging (20),

$$\text{MAPS} = \frac{P_{e,T}}{P_{\bar{u},T}} V_c \tag{21}$$

As an example, say a member of a T-node group of 10 wishes to setup a multicast group in a narrowband network where the average bit-error-rate is 10^{-3}. Figure 41 (or equation (19)) indicates that there is 99% chance that the request-reject procedure succeeds, i.e., the probability of reusing a group address because the sender did not receive any REJECT PDU is 1%. Being a relatively small group, the sender estimates that no more than 3 messages are typically transmitted at once over the network. In a narrowband network, the probability of non-uniqueness in the ownership of the multicast group should be low to prevent unnecessary network loading. A typical value might be 10^{-6} or less, which in the case being discussed would set the multicast address pool size to:

$$\text{MAPS} \geq \frac{10^{-2}}{10^{-6}} 3 \tag{22}$$

i.e. 30 000 addresses or more. Note that setting $P_{\bar{u},T}$ to 10^{-6} does not guarantee that the accidental sharing of a same multicast address will occur at most once in a million. The probability of a multicast address being shared is determined by the ratio V_c/MAPS, which in this example is 1 in 10 000. $P_{\bar{u},T}$ rather indicates that there is one chance in one million that these accidental allotments will not be detected by the sender when trying to establish a new multicast group.

6.0 Summary

P_MUL is able to provide reliable multicast message delivery under a variety of network conditions. The efficiency it is able to achieve varies greatly depending on the nature of the underlying network. This section summarizes several of the problems and observations made during this study.

1. Vulnerability to the loss of control packets — P_MUL is unable to recover gracefully from the loss of control packets. In noisy environments, the loss of control packets is not an uncommon occurrence. P_MUL is able to recover from the loss of most control packets, but is vulnerable to the loss of ADDRESS_PDUs. ADDRESS_PDUs lost when the receivers are expecting responses to their final acknowledgements may cause the message transfer to terminate incorrectly. Similarly, a receiver can mistakenly believe that it has received the final ADDRESS_PDU when in fact it has not. These situations are not rare, and when they occur the network is polluted with a large number of unneeded packets, in some cases indefinitely. It is recommended that the ADDRESS_PDU detection scheme be made more robust, perhaps by numbering the complete set of ADDRESS_PDUs, in order to reduce the probability of faulty termination.

2. Pollution of the network — When a transmitter sends a message at a much higher rate than that allowable by a subnetwork on which resides some receiver in the multicast group, P_MUL can create a large amount of traffic that will pollute the network long after the message has been successfully received and acknowledged. In order to avoid this type of situation better flow control between the transmitter and the receivers is needed. Congestion avoidance and congestion control (CA/CC) were not explicitly addressed by the ACP 142. The Internet-Draft briefly mentions two CA/CC schemes, which were both partly investigated in the study. The use of the PDU_Delay parameter to delay the transmission of individual packets was found to be effective in reducing the number of unnecessary packets transmitted by the sender. However, it is not clear if the scheme is able to avoid any conflict with other CA/CC mechanisms, which could result in improper feedback signal to control the traffic flow. Also, it would be important to specify the scheme in more details, e.g., with regard to traffic priority. The use of the ACK Re-transmission timer to progressively reduce the number of retransmissions was also found to be an effective flow control mechanism, at least in noise-free environments. The ACP 142 indicates that the ACK Re-transmission timer should be initialised with the value of the *round trip delay*[1] plus a safeguard but the simulation results indicate that the initialisation value should in fact be selected according to the expected link quality. For instance, in an error-free environment, best results are achieved when the timer duration is set high, i.e. to the time it takes to transmit the entire message once and to receive the corresponding acknowledgements. The timer can be progressively and incrementally increased if heavy traffic is expected. In noisy environments, the initialisation value of the ACK Re-transmission timer should be low (i.e., set to the time it takes for a PDU of maximum size to travel the distance sender-receivers-sender) and kept low, otherwise, fewer retransmissions can occur and the message transfer may never be completed before the message expiry time has been reached.

1. A term that should be explicitly defined in future releases of the ACP 142 to clarify its meaning.

3. Lack of feedback — In an error-free network, P_MUL is a very quiet and efficient protocol. Each receiver need send only one packet to acknowledge the entire message, no matter how long that message may have been. Similarly, the overhead required for the sender is also small – a series of ADDRESS_PDUs to begin the transmission and a single ADDRESS_PDU to end it. But, most real networks are not error-free, and the silence of the protocol means that the loss, or untimely arrival, of a single control packet can be significant. In order to improve the flow control between the transmitter and the receivers, it would be advisable to increase the amount of feedback to the sender even under relatively noiseless conditions.

4. ACK implosion problem — Although the number of ACK_PDUs produced by the protocol can be very small, ACK implosion at the sender may be a problem, especially for large multicast group sizes, when a group of receivers attempts to send ACK_PDUs simultaneously. The ACK_PDU_Jitter timer has been suggested as a method of avoiding ACK implosion. However, it was found that the ACK_PDU_Jitter timer could reduce the throughput achievable when sending many small messages in succession. Unless the multicast group is extremely small, the message transfer will always be delayed by an amount close to the maximum jitter value of the jitter window. It may be worthwhile to investigate adjusting the length of the ACK_PDU_Jitter timer to suit, among other things, the length of the message to be sent. Another avenue might be to have the receivers send acknowledgements more often than is currently allowed by the protocol, in particular during the course of the first transmission (see the *Lack of feedback* problem mentioned in item 3) where receivers could take turns, perhaps sequentially or randomly, in informing the sender of the state of the message transfer at each destination. The likelihood of ACK implosion at the sender and the effectiveness of the ACK_PDU_Jitter timer as a method of avoiding it needs to be further investigated.

5. Message transfer under noisy conditions — P_MUL's performance deteriorates rapidly with increasing bit error rate. For a maximum PDU size of 1024 octets, message transfers start to fail around 10^{-3} errors per bit. Simulation results show that this limit can be pushed up by an order of magnitude when the maximum PDU size is small (around 64 octets). Also, the protocol's performance remains quite constant as long as the value of the M_Missing_DATA_PDUS parameter is set to 6 PDUs or more. The use of erasure coding helps improve the message delivery time and throughput but a $(3k/2, k)$ code appears to be just as effective as a $(2k,k)$ code. Coding is most effective when the transmission errors over the network are not too numerous. As the bit error rate increases (e.g. from 10^{-3} errors per bit) P_MUL gradually produces a large number of ACK_PDUs and makes many retransmissions, neither of which will ever be sufficient to ensure successful message delivery. In fact, although the total number of ACK_PDUs received does increase at high bit error rates, there comes a point where (around 5×10^{-3}) the number of ACK_PDUs received in error outnumbers the number of ACK_PDUs successfully received, suggesting that further increasing the number of ACK_PDUs will have little effect. When time is a constraint, P_MUL is rather limited in its capability of transferring large files when the transfer must be done in a noisy environment. In all fairness though, it would be necessary to compare P_MUL to other reliable transport protocols to better assess the performance obtained in similar conditions.

6. Dynamic Group Installation — This study derived some analytical expressions showing that the *Silent Procedure* used by P_MUL to set up multicast groups is relatively robust and does not require a very large pool of multicast addresses. However, the dynamic

address negotiation process can be costly for two reasons. First, the process introduces overhead independent of the length of the message. For short messages, the time required to negotiate a multicast address can be much greater than the length of time required to transfer the message. Secondly, the negotiation is done on a per message basis. If the sender needs to send thousands, hundreds, or perhaps even tens of short messages then dynamic group installation is a convenience that can be costly.

7. Multicast Group Size — The results obtained with group sizes up to forty-two show a pattern that may well extend to larger group sizes. The results suggest that group size does not impact significantly the P_Delivery Time or throughput achievable by P_MUL. However, with increasing group size ACK implosion and the loss of significant control packets could become problematic issues.

7.0 Conclusion

The P_MUL protocol was designed for use in narrowband military networks where some nodes may be subject to EMCON conditions. The results of this simulation study indicate that for well-behaved senders, P_MUL uses bandwidth very efficiently in heterogeneous noise-free environments. However, the study also was able to uncover weaknesses in the protocol that may cause problems in other network environments.

The simulation results obtained suggest areas where further study, and in some cases improvements to the protocol, may be needed. These include:

1. Congestion avoidance and congestion control – P_MUL is likely to pollute a heterogeneous network environment with a large number of unnecessary packets if no CA/CC scheme is implemented. Several possible CA/CC schemes are available to P_MUL. It would be interesting to implement a more dynamic form of CA/CC in the model to study its impact on P_MUL's performance.
2. Flow Control – P_MUL produces very few ACK_PDUs in relatively noise-free environments. It might be possible to improve P_MUL's performance by increasing the amount of feedback to the sender, perhaps through the periodic transmission of positive ACK_PDUs, without consuming an unacceptable amount of additional bandwidth.
3. Robustness – P_MUL is vulnerable to the loss of control packets, particularly final ADDRESS_PDUs. A scheme to improve the robustness of control packet transmission, perhaps through more stringent accounting of received packets, or through the retransmission of crucial control packets, should be investigated.
4. ACK implosion – This study did not investigate the problem of ACK implosion at the sender. Since reliable multicast protocols are prone to ACK implosion, it has been suggested that a random timer be used to delay the transmission of ACK_PDUs from the receivers. Further study is required to determine the effectiveness of this timer in avoiding ACK implosion.
5. Reliability – The forward error correction scheme used in this study provided unconvincing performance improvement at high bit error rates (above 5×10^{-3}). It would be necessary to compare P_MUL to other reliable transport protocols to better assess the performance obtained in similar conditions. The use of different error correction algorithms should be tested to determine if they are better able to improve P_MUL's performance at high bit error rates.

P_MUL has acheived its goal of providing reliable multicast transmission of messages. Its relatively quiet operation makes it suited for low-bandwidth network environments. However, improvements to the protocol need to be made to address the issues raised in this simulation study.

8.0 References

[1] CCITT (1988). *Data Communication Networks: Message Handling Systems, Recommendations X.400 - X.420*, ISO Blue Book.

[2] Poste, J. (1982). *Simple Mail Transfer Protocol*, RFC 821.

[3] CSNI Technical Group Members (1996). *CSNI-2 Final Report*, Document NATO CSNI/2/D/003/Rev1.

[4] Allied Communication Publication (1999). *ACP 142, P_MUL: An Application for Multicast Messaging under EMCON Restriction*, ACP 142 Draft 0.2a.

[5] Riechmann, C. (1999). *P_Mul: An Application Protocol for the Reliable Data Transfer over Multicast Subnetworks and under EMCON Restrictions*, Internet Draft (Working document of the Internet Engineering Task Force- IETF).

[6] Rizzo, L. (1997). *Effective Erasure Codes for Reliable Computer Communication Protocols*, acm sigcomm, Computer Communication Review, Vol. 27, No.2

[7] Li D., and R. Cheriton (1999). Evaluating the Utility of FEC with Reliable Multicast, Proceedings of the 1999 International Conference on Network Protocols, Toronto, Canada, pp. 97-105

[8] Riechmann, C. (1999). Private communications.

9.0 Acronyms and Initialisms

ACP	Allied Communication Publication
BER	Bit Error Rate
CA/CC	Congestion Avoidance and Congestion Control
CCEB	Combined Communications Electronic Board
CCITT	International Telegraph and Telephone Consultative Committee
DVMRP	Distance Vector Multicast Routing Protocol
EMCON	Emission Control
ICMP	Internet Control Message Protocol
IP	Internet Protocol
ISO	International Organization for Standardization
MAP	More ADDRESS_PDUs
MAPS	Multicast Address Pool Size
MHS	Message Handling System
MMHS	Military Message Handling System
MOSPF	Multicast Open Shortest Path First
MPDU	Message Protocol Data Unit
P1	Message Transfer Protocol
PDU	Protocol Data Unit
PIM	Protocol Independent Multicast
RFC	Request for Comment
RMP	Reliable Multicast Protocol
SMTP	Simple Mail Transfer Protocol
STANAG	(NATO) Standardisation Agreement
UDP	User Datagram Protocol

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATOR (the name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Establishment sponsoring a contractor's report, or tasking agency, are entered in section 8.)

COMMUNICATIONS RESEARCH CENTRE
3701 CARLING AVENUE, P.I. BOX 11490, STN H
OTTAWA, ONTARIO, CANADA, K2H8S2

2. SECURITY CLASSIFICATION (overall security classification of the document, including special warning terms if applicable)

UNCLASSIFIED

3. TITLE (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S,C or U) in parentheses after the title.)

A SIMULATION OF P_MUL (AN APPLICATION FOR MULTICAST MESSAGING UNDER EMCON RESTRICTION) (U)

4. AUTHORS (Last name, first name, middle initial)

Bilodeau, Claude

Dumoulin, Sarah

5. DATE OF PUBLICATION (month and year of publication of document)	6a. NO. OF PAGES (total containing information. Include Annexes, Appendices, etc.)	6b. NO. OF REFS (total cited in document)
June 2000	xiii + 102	8

7. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

TECHNICAL REPORT

8. SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development. Include the address.)

DEFENCE RESEARCH ESTABLISHMENT OTTAWA (DREO)
3701 CARLING AVENUE, OTTAWA, ONTARIO
K1A0Z4

9a. PROJECT OR GRANT NO. (if appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant)	9b. CONTRACT NO. (if appropriate, the applicable number under which the document was written)
2KN14	

10a. ORIGINATOR'S DOCUMENT NUMBER (the official document number by which the document is identified by the originating activity. This number must be unique to this document.)	10b. OTHER DOCUMENT NOS. (Any other numbers which may be assigned this document either by the originator or by the sponsor)
	CRC Report No. 2000-005 DREO Report No. DREO TR2000-048

11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

(X) Unlimited distribution
( ) Distribution limited to defence departments and defence contractors; further distribution only as approved
( ) Distribution limited to defence departments and Canadian defence contractors; further distribution only as approved
( ) Distribution limited to government departments and agencies; further distribution only as approved
( ) Distribution limited to defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in 11) is possible, a wider announcement audience may be selected.)

UNLIMITED

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

P_MUL is an application layer reliable multicast protocol specifically designed for use over narrowband networks where nodes may be operating in EMCON (Emission Control) or Radio silence mode. The protocol is specified in the Allied Communication Publication (ACP) 142. The Communications Research Centre (CRC) has created a discrete-event simulation model of P_MUL in order to characterise its performance under a variety of operating conditions. Simulation scenarios were created to examine the effectiveness of P_MUL in both noise-free and noisy channel environments. The scenarios focus on the impact of protocol parameters on the results obtainable by P_MUL.

This report describes the CRC model of P_MUL and outlines the simulation scenarios studied. Analysis of simulation results is provided, weaknesses of the protocol are identified, and optimal values for certain protocol parameters are suggested.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

P_MUL
Multicast
Reliable multicast
Simulation
EMCON
Radio silence
Erasure Coding

Defence R&D Canada
is the national authority for providing
Science and Technology (S&T) leadership
in the advancement and maintenance
of Canada's defence capabilities.

R et D pour la défense Canada
est responsable, au niveau national, pour
les sciences et la technologie (S et T)
au service de l'avancement et du maintien des
capacités de défense du Canada.

www.drdc-rddc.dnd.ca